Jovis Relational Commands

Syntax Reference

AppendSelection

SYNTAX

```
Get Jovis (
[1] "AppendSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Criteria", ¬
[5] '[Function Name]', ¬
[6] '[Prompt]', ¬
[7] '[Selection Max]', ¬
[8] '[Continue Flag]' )
```

WHERE

[1] is the Jovis 'AppendSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the criteria for appending selections;
[5] is an optional parameter for a function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[6] is an optional parameter for the 'prompt' progress dialog;
[7] is an optional parameter for the maximum number of records to be selected.
[8] is an optional "True" or "False" parameter. If "False" or missing, the given criteria replaces the previous criteria and appends all records previously selected, checking for duplicate records in the existing selection, and replacing any that are contained in the selection. If this flag is "True", the 'current' criteria is used, and further appending continues.

EXAMPLE

```
on DemoScript
  Put "Field zip = " & quote & "60612" & quote into theCriteria
  get Jovis("AppendSelection", "myDB","Customers",theCriteria)
end DemoScript
```

DESCRIPTION

Adds records that meet the given criteria to a current selection.

COMMENTS

Checks for duplication of records in the existing selection, and replaces any that are already contained in the selection.

SEE ALSO

SetSelection
Custom Function Handlers
TrimSelection

BeginTransaction

SYNTAX

```
Get Jovis (
[1] "BeginTransaction",¬
[2] "FileGlobal" )
```

WHERE

[1] is the Jovis 'BeginTransaction' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
on DemoScript
  get Jovis("BeginTransaction", "myDB")
end DemoScript
```

DESCRIPTION

Once this command is called it causes records read or created to be locked. Never leave records locked for an extended length of time. Other clients may need to make changes or access the most recent update. The maximum number of records that can be locked is 200 per 5 client server (40 records per user); 400 per 10 client server, etc.

COMMENTS

If a script needs to lock records contained in more than one database, then it must execute transaction commands for each database involved.

If scripted for the single-user, this command does nothing. By scripting this command into the single-user version you will be able to switch to the client/server environment without modifying your scripts.

This command is non-functional in the single-user version.

SEE ALSO

CommitTransaction
CancelTransaction
IsTransactionOn

CancelTransaction

SYNTAX

```
Get Jovis (
[1]"CancelTransaction",¬
[2] "FileGlobal", )
```

WHERE

[1] is the Jovis 'CancelTransaction' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
on DemoScript
  get Jovis("CancelTransaction", "myDB")
end DemoScript
```

DESCRIPTION

Throws away all updates to the database and unlocks the appropriate records. Once this command is called it causes records read or created to be locked. Never leave records locked for an extended length of time. Other clients may need to make changes or access the most recent update. The maximum number of records that can be locked is 200 per 5 client server (40 records per user); 400 per 10 client server, etc.

COMMENTS

This command is non-functional in the single-user version of Jovis.

If a script needs to lock records contained in more than one database, then it must execute transaction commands for each database involved.

If scripted for the single user, this command does nothing. By scripting this command into the single-user version you will be able to switch to the client-server environment without modifying your scripts.

SEE ALSO

CommitTransaction
BeginTransaction
IsTransactionOn

ClearCriteria

SYNTAX

```
Get Jovis (
[1] "ClearCriteria", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'ClearCriteria' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;

EXAMPLE

```
on DemoScript
  get Jovis("ClearCriteria", "myDB","Customers")
end DemoScript
```

DESCRIPTION

Removes the existing criteria without changing the position in the database. This command is used in conjunction with NextRecord. It allows you to continue stepping to the next record without any criteria.

SEE ALSO

Record Paths
NextRecord
PriorRecord
GetRecordCriteria

CloseCollection

SYNTAX

```
Get Jovis (
[1] "CloseCollection", ¬
[2] "FileGlobal" )
```

WHERE

[1] is the Jovis 'CloseCollection' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
on DemoScript
  get Jovis("CloseCollection", "myDB")
end DemoScript
```

DESCRIPTION

Closes the database file.

COMMENTS

Closes the database file and deallocates all memory used while it was open.

In the client/server version, this command closes the client’s channel to the database, and deallocates all memory used by the client.

SEE ALSO

ShutDown

CommitTransaction

SYNTAX

```
Get Jovis (
[1] "CommitTransaction", ¬
[2] "FileGlobal" )
```

WHERE

[1] is the Jovis 'CommitTransaction' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
on DemoScript
  get Jovis("CommitTransaction", "myDB")
end DemoScript
```

DESCRIPTION

Applies all updates to the database and unlocks all records previously locked by the client.

COMMENTS

This command is non-functional in the single-user.

SEE ALSO

IsTransactionOn
UpdateRecord
CancelTransaction
BeginTransaction

CountMatches

SYNTAX

```
Put Jovis (
[1] "CountMatches", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Criteria", ¬
[5] '[Validate Only]', ¬
[6] '[Function Name]', ¬
[7] '[Prompt]' )
```

WHERE

[1] is the Jovis 'CountMatches' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the a standard criteria that evaluates to true if there is a match
[5] If Validate Only is set to true, then no comparisons are performed. The only action is to validate the given criteria . Check the JovisErrorCode to see if there were any errors in the criteria. See Error and Warning Handlers on how to keep this evaluation local to your handler.
[6] is an optional parameter function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[7] is an optional parameter for a 'prompt' for the progress dialog.

EXAMPLE

```
on DemoScript
  put "Field LastName starts_with [jack]" into criteria
  put Jovis("CountMatches","myDb","Customers",criteria)into matchCount
end DemoScript
```

DESCRIPTION

Counts the number of records in the relation that match the criteria. This does not create or affect an active selection. If you simply need the number of records in a given relation, use the command 'CountRelation'.

COMMENTS

This command will read the same number of records, depending on the criteria, as the SetSelection command. Use this command when you definitely do not need a selection.

SEE ALSO

SetSelection
CountRelation
CountSelection

CountRelation

SYNTAX

```
Put Jovis (
[1] "CountRelation", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'CountRelation' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;

EXAMPLE

```
on DemoScript
  put Jovis("CountRelation","myDB","Customers")   into recordCount
end DemoScript
```

DESCRIPTION

Returns the number of records in a given relation.

SEE ALSO

UpdateRecord

CountSelection

SYNTAX

```
Put Jovis (
[1] "CountSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Named Selection]' )
```

WHERE

[1] is the Jovis 'CountSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional parameter for a named selection rather than the current selection.

EXAMPLE

```
on DemoScript
  put Jovis("CountSelection","myDB","Customers") into tableSize
end DemoScript
```

DESCRIPTION

Returns the number of records in the current selection or a named selection for the given relation. If you simply need the number of records in a relation, use 'CountRelation'.

SEE ALSO

Multiple Selection CountMatches
SetSelection

CreateCollection

SYNTAX

```
Get Jovis (
[1] "CreateCollection", ¬
[2] "FileGlobal",  ¬
[3] '[Complete File Path]', ¬
[4] '[Prompt]', ¬
[5] '[File Dlg Coords]', ¬
[6] '[Creator,[File] Types]', ¬
[7] '[Read-Only Password]', ¬
[8] '[Read-Write Password]' )
```

WHERE

[1] is the Jovis 'CreateCollection' command;
[2] is the global identifier for the file you want to work with;
[3] is the optional file path to create the data file. If this parameter is empty, the standard put file dialog will be displayed;
[4] is an optional prompt, used with the standard put file dialog;
[5] is the optional coordinates for displaying the standard put file dialog;
[6] is the optional creator and file types to assigned to the data file. For example: "DASW,PROR" — which are also the defaults; This parameter will override any input from the standard put file dialog.(FileType can only be set from this parameter.)
[7] is an optional parameter for assigning a Read-Only password to the data file.
[8] is an optional parameter for assigning a Read-Write password to the data file.

EXAMPLE

```
on DemoScript
  global myDB
  if myDB is empty then
    put "Jovis" into myDB
    get Jovis("CreateCollection", "myDB")
    put "ReadMe" into ROPassword
    put "Kukamonga" into RWPassword
    get Jovis("CreateCollection "myDB","","What do you want to call the
         file?",¬
    "90,120","MYDB,txt2",ROPassword,RWPassword)
    get Jovis("CreateCollection", "myDB","HD20:my database file")
    -- Client/Server:
    get Jovis("CreateCollection", "myDB","my dbfile:myserver@*")
  end if
end DemoScript
```

DESCRIPTION

Creates a database file. If the file does not exist, it creates a new one.

COMMENTS

A shell application Global must be initialized to “Jovis” before using this command. CreateCollection will update this global to it's required information.

You can only over-write a data file via the standard put file dialog. Attempting to do so, using the 3rd parameter will cause an error.

All error reporting is done using the "JovisErrorCode" global.

SEE ALSO

OpenCollection
DBInit
OpenDB
Exclusive Status
Initializing a Jovis Global

CreateField

SYNTAX

```
Get Jovis (
[1] "CreateField", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field Name", ¬
[5] '[Field Type]' )
```

WHERE

[1] is the Jovis 'CreateField' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the field being created;
[5] is an optional parameter for indicating the field's type. There are four available types: Number, Text, Date, and Logic. The default is 'Text".

EXAMPLE

```
on DemoScript
  get Jovis("CreateField", "myDB","Customers","order_number","number")
  get Jovis("CreateField", "myDB","Customers","description","text")
  get Jovis("CreateField", "myDB","Customers","dateShipped","date")
  get Jovis("CreateField", "myDB","Customers","back_ordered","logic")
end DemoScript
```

DESCRIPTION

Creates a field in the given relation.

COMMENTS

There are four valid field types: Number, Text, Date, and Logic (i.e. Bodean).

SEE ALSO

CreateRelation
CreateIndex
Exclusive Status
CreateCollection
OpenCollection
Valid Names

CreateIndex

SYNTAX

```
Get Jovis (
[1] "CreateIndex", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "FieldName", ¬
[5] '[KeyLength]' )
```

WHERE

[1] is the Jovis 'CreateIndex' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the field to be indexed;
[5] if the field type is 'TEXT', you should indicate how many characters of the field to have indexed. This parameter is ignored if the field type is not 'TEXT'. If the field type is 'TEXT' and this parameter is empty, or missing the key length is 12.

EXAMPLE

```
on DemoScript
  get Jovis("CreateIndex", "myDB","Customers","Last_Name","8")
  get Jovis("CreateIndex", "myDB","Customers","Account_Start")
  get Jovis("CreateIndex", "myDB","Customers","orderNbr")
end DemoScript
```

DESCRIPTION

Creates an index for the field indicated.

COMMENTS

Jovis considers an empty field in a record to be undefined. It will not attempt to index null values (i.e. empty fields). This means that you cannot search an index with criteria such as "Field LastName = " " and expect to find all records with no LastName. Instead, this criteria would return a warning stating that no records were selected. If you need to search an index for empty records, you should initialize the field for each record to a predetermined value, such as "**EMPTY**". (If the field type is NUMBER, the predetermined value should be zero. This allows the 'SelectionStats' command to work as expected.) A date field might be initialized to perhaps 1/1/1904 as a default starting date.

SEE ALSO

CreateField
CreateRelation
CreateCollection
Exclusive Status
DefineKeyset
DelIndex
Valid Names

CreateRecord

SYNTAX

```
Put Jovis (
[1] "CreateRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'CreateRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the new relation you want to work within the designated file;

EXAMPLE

```
on DemoScript
  put Jovis("CreateRecord","myDB","Customers")   into newOrderRec
end DemoScript
```

DESCRIPTION

Returns a new empty record for the given relation.

COMMENTS

Once this command completes, all the fields are empty, except the !RecID field which is automatically assigned by Jovis. Always use the 'SetRecordField' and 'GetRecordField' to access the record's fields. Never insert or retrieve data directly from a record.

SEE ALSO

SetRecordField
UpdateRecord
DelRecord
BeginTransaction
CommitTransaction

CreateRelation

SYNTAX

```
Get Jovis (
[1] "CreateRelation", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'CreateRelation' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the new relation you want to create within the designated file;

EXAMPLE

```
on DemoScript
  get Jovis("createRelation", "myDB","Customers")
end DemoScript
```

DESCRIPTION

Creates an empty relation in the given data file using the name passed as the second parameter.

COMMENTS

If the collection already has a relation with that name, an error is returned.

SEE ALSO

CreateCollection
OpenCollection
Open Exclusive
CreateField

CurrentRecord

SYNTAX

```
Put Jovis (
[1] "CurrentRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[RecID Only]' )
```

WHERE

[1] is the Jovis 'CurrentRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional parameter. If "True", returns only the record's ID. If it is false or missing, the entire record is returned. This is useful in conjunction with the DelByRecID command, which allows you to delete locked records (under Jovis client/server) by passing the !RecID value instead of the entire record.

EXAMPLE

```
on DemoScript
  put Jovis("CurrentRecord","myDB","Customers") into Rec
end DemoScript
```

DESCRIPTION

Returns the current record for the active record path.

SEE ALSO

Record Paths Tutorial
DelByRecID

DelByRecID

SYNTAX

```
Get Jovis (
[1] "DelByRecID", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record ID" )
```

WHERE

[1] is the Jovis 'DelByRecID' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the Record ID for record you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("DelByRecID", "myDB","Customers","1471")
end DemoScript
```

DESCRIPTION

This command works the same as DelRecord. The record is deleted from the relation, and removed from all the indexes. The difference is that here you do not need to pass the entire record as a parameter, in order to delete it. (This may be particularly useful if you are using the multi-user version, and your records are large and time-consuming to send across the network.) Use GetRecordField to retrieve the record ID from the "!RecID" field for the record you want deleted.

COMMENTS

In the multi-user version, changes made with DelByRecID are stored at the Server and the collection is not physically changed until the CommitTransaction command is executed.

SEE ALSO

CommitTransaction
UpdateRecord
CancelTransaction

DelField

SYNTAX

```
Get Jovis (
[1] "DelField", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field Name" )
```

WHERE

[1] is the Jovis 'DelField' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the field you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("DelField", "myDB","Customers","on_hand")
end DemoScript
```

DESCRIPTION

Removes the field from the list of valid field names in the specified relation.

COMMENTS

This process is not reversible.

The records in the relation are not changed to remove the data that is contained in the deleted field. However, it is flagged as no longer used.

If the field is indexed, the index must be deleted before the field can be deleted.

SEE ALSO

DelIndex
Exclusive Status
DelRelation

DelIndex

SYNTAX

```
Get Jovis (
[1] "DelIndex", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field Name" )
```

WHERE

[1] is the Jovis 'DelIndex' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of field for the index you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("DelIndex", "myDB","Customers","orderNumber")
end DemoScript
```

DESCRIPTION

Deletes a given index in the specified relation.

COMMENTS

This process is not reversible.

SEE ALSO

Exclusive Status
DelRelation

DelRecord

SYNTAX

```
Get Jovis (
[1] "DelRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record" )
```

WHERE

[1] is the Jovis 'DelRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the entire Jovis record you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("delRecord", "myDB","Customers",orderRecord)
end DemoScript
```

DESCRIPTION

Deletes the record from the relation and removes the field data from all the indexes for the record's relation. For the multi-user version, changes made with DelRecord are stored at the Server and the database is not physically changed until the CommitTransaction command is executed.

COMMENTS

DelRecord will actually delete the record from the database file. Do not follow DelRecord with the UpdateRecord command.

SEE ALSO

DelByRecID
BeginTransaction
CommitTransaction
CancelTransaction

DelRecordPath

SYNTAX

```
Get Jovis (
[1] "DelRecordPath", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record Path Name" )
```

WHERE

[1] is the Jovis 'DelRecordPath' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the record path you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("DelRecord","myDB","Customers","MyRecordPath")
end DemoScript
```

DESCRIPTION

Removes the named record path from the list of record paths, and frees the memory it was using.

SEE ALSO

Record Paths
ListRecordPaths
ReserveRecordPath

DelRelation

SYNTAX

```
Get Jovis (
[1] "DelRelation", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'DelRelation' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("DelRelation", "myDB","Customers")
end DemoScript
```

DESCRIPTION

Deletes an entire relation and all the records in it, including its indexes. This command is not reversible!

COMMENTS

While the relation is being deleted, a dialog box is displayed showing its progress. You can stop the process by clicking on the cancel button.
At that point the relation is flagged as invalid. You can no longer use the relation. Your only option is to finish deleting it at a later time. Quite often it's easier and faster to create a new data file than deleting an entire relation. See 'ExportData' and 'ImportData'.

SEE ALSO

DelIndex
DelField
CreateCollection
DBInit

DelSelection

SYNTAX

```
Get Jovis (
[1] "DelSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Selection Name" )
```

WHERE

[1] is the Jovis 'DelSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the Selection you wish to delete.

EXAMPLE

```
on DemoScript
  get Jovis("DelSelection", "myDB","Customers","TempSelection")
end DemoScript
```

DESCRIPTION

Removes the named selection from the list of selections, and frees the memory it was using.

SEE ALSO

Multiple Selection
ListSelections
ReserveSelection

DupRecordPath

SYNTAX

```
Get Jovis (
[1] "DupRecordPath", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record Path Name" )
```

WHERE

[1] is the Jovis 'DupRecordPath' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of a Record Path you wish to duplicate.

EXAMPLE

```
on DemoScript
  get Jovis("DupRecordPath", "myDB","Customers","FirstRecordPath")
end DemoScript
```

DESCRIPTION

Makes a copy of the active record path, and reserves it in memory, identified by the name passed as the third parameter.

COMMENTS

When this command finishes, the active record path is unaffected. But there is a copy of it in the list of reserved record paths of the given relation.

SEE ALSO

Record Paths
ListRecordPaths
ReserveRecordPath

DupSelection

SYNTAX

```
Get Jovis (
[1] "DupSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Selection Name" )
```

WHERE

[1] is the Jovis 'DupSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name to use for the duplicate selection.

EXAMPLE

```
on DemoScript
  get Jovis("DupSelection", "myDB","Customers","MasterSelection")
end DemoScript
```

DESCRIPTION

Makes a copy of the current selection for the given relation using the supplied name in parameter four, and reserves it in memory.

COMMENTS

When this command finishes, the active selection is unaffected. But there is a copy of it in the list of reserved selections for the indicated relation.

SEE ALSO

Multiple Selection
ListSelections
ReserveSelection
DelSelection

ExportData

SYNTAX

```
Get Jovis (
[1] "ExportData", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field List", ¬
[5] '[Criteria]', ¬
[6] '[Field Delim]', ¬
[7] '[Record Delim]', ¬
[8] '[File Path]', ¬
[9] '[SFPrompt]', ¬
[10] '[Function Handler]', ¬
[11] '[Progress Prompt]' )
```

WHERE

[1] is the Jovis 'ExportData' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is a comma delimited list of the names of the fields in the relation you want exported. The field(s) in the search criteria need not be included in the field list. In other words, you can select on one set of fields and export a different set;
[5] is the criteria you want to base your export on, the default is to have all records exported;
[6] is the field delimiter you want to use, the default is a tab;
[7] is the the record delimiter to use, the default is a carriage return;
[8] is the entire file path to the location of the file you want to create, the default is to display the standard file put dialog;
[9] is the standard file put dialog prompt;
[10] is an optional parameter for a function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[11] is the prompt for the progress dialog.

EXAMPLE

```
on DemoScript
  get "Field paidFlag = " & quote & false  & quote into criteria
  get "orderNum,customer,name,address,city,state,zip" into fList
  get Jovis("ExportData", "myDB","Customers",fList, criteria,
         return,"HD20:orders:unpaid  Type option "2" orders")
end DemoScript
```

DESCRIPTION

Exports data from the database to a text file.

COMMENTS

The records are exported in the order that they are found in the database.   If it is essential that the records be exported in a particular order, use the selection commands, SetSelection, SortSelection and ExportSelection.

SEE ALSO

ImportData
ExportSelection

ExportSelection

SYNTAX

```
Get Jovis (
[1] "ExportSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Field Delim]', ¬
[5] '[Record Delim]', ¬
[6] '[File PathName]', ¬
[7] '[FSDialog Prompt]', ¬
[8] '[Field List]', ¬
[9] '[Record Range]', ¬
[10] '[Progress Prompt]', ¬
[11] '[Named Selection]' )
```

WHERE

[1] is the Jovis 'ExportSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the field delimiter to be used. The default is a tab character;
[5] is the record delimiter to be used. The default is a carriage return character;
[6] is the entire file path name. If it is empty, the standard put file dialog is displayed.
[7] is the prompt for the standard put file dialog;
[8] is a comma delimited list of fields to be exported for each row of the selection. This is also the order in which the fields are to appear in the text file. This works equally well with all the fields, or a subset of the fields in the selection.
[9] refers to the rows in the selection. If you do not wish to use all the records in the selection, you may specify a range of record numbers instead. If you give one number for the parameter, then the record range will be from the first selection record up to the number you have given. If you give two numbers separated by a comma, the record range will be all the records in that range, including the two given as parameters. If the parameter is empty or missing, all the records in the selection are exported;
[10] is the progress dialog prompt;
[11] is a named selection for the given relation rather than the active selection.

EXAMPLE

```
on DemoScript
  get Jovis("ExportSelection", "myDB","Customers")
  put "First_Name,Last_Name,Address1,Address2,State,Zip" into fldList
  get Jovis("ExportSelection", "myDB","Customers","","","","", ¬
                fldList,"5,12")
end DemoScript
```

DESCRIPTION

Exports data from a selection to a text file.

COMMENTS

The exported file uses the 'Teach Text' file type.

SEE ALSO

SetSelection
SortSelection
AppendSelection
TrimSelection
Multiple Selections

FillHCButtons

SYNTAX

```
Get Jovis (
[1] "FillHCButtons", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Field List", ¬
[6] "Button List", ¬
[7] '[Card/Bkgnd layer]' )
```

WHERE

[1] is the Jovis 'FillHCButtons' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the record used as the source for setting card or bkgnd buttons;
[5] is a comma delimited list of record fields;
[6] is a comma delmited list of card or bkgnd buttons as the destination;
[7] is an optional flag indicating whether the buttons are in the card or bkgnd layer. If this parameter is "false" or missing, the bkgnd layer is used.

EXAMPLE

```
on DemoScript
  Put "Married,over50,Children"  into FieldList
  Put "isMarried,isOver50,hasChildren"  into HCList
  get Jovis("FillHCButtons",
         "myDB","Customers",rec,FieldList,HCList,False)
end DemoScript
```

DESCRIPTION

Sets the hilite of buttons based upon the 'True' or 'False' value in the record's fields.

COMMENTS

This command matches the Jovis fields with buttons by their relative position in the list (i.e., item 1 of record field list to item 1 of the button list).

If any of the Jovis record fields do not exist, an error message is issued and the command is not executed.

If any of the buttons do not exist, a warning is issued and the hilite is set for as many buttons as possible.

SEE ALSO

FillRecButtons

FillHCFields

SYNTAX

```
Get Jovis (
[1] "FillHCFields", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Record Field List", ¬
[6] "User Field List", ¬
[7] '[Card/Bkgnd Field]' )
```

WHERE

[1] is the Jovis 'FillHCFields' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the record used as the source for setting card or bkgnd fields;
[5] is the a comma delimited list of record fields;
[6] is a comma delmited list of card or bkgnd fields used as the destination;
[7] is an optional flag indicating whether the fields are in the card or bkgnd layer. If this parameter is "false" or missing, the bkgnd layer is used.

EXAMPLE

```
on DemoScript
  Put "last_name,first_name,address" into FieldList
  Put "Last name,first name,address" into HCList
  get Jovis("FillHCFields", "myDB","Customers",rec,FieldList,HCList,False)
end DemoScript
```

DESCRIPTION

Takes a Jovis record and populates the shell application's fields with the field data from the record.

COMMENTS

This command matches the database fields with the shell application fields by their relative position in the list (i.e., item 1 of record field list to item 1 of the shell's field list).

If any of the Jovis fields do not exist, an error message is issued and the command is not execut-ed.

If any of the fields do not exist, a warning is issued and as many fields as possible are filled on the card.

SEE ALSO

FillRecFields

FillRecButtons

SYNTAX

```
Put Jovis (
[1] "FillRecButtons", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Field List", ¬
[6] "Buttons List", ¬
[7] '[Card/Bkgnd flag]' )
```

WHERE

[1] is the Jovis 'FillRecButtons' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the record used as the destination of the card or bkgnd buttons;
[5] is a comma delimited list of record fields;
[6] is a comma delmited list of card or bkgnd buttons used as the source for setting the Jovis record;
[7] is an optional flag indicating whether the buttons are in the card or bkgnd layer. If this parameter is "false" or missing, the bkgnd layer is used.

EXAMPLE

```
on DemoScript
  put "Married,over50,Children"  into FieldList
  Put "isMarried,isOver50,hasChildren" into HCList
  Put Jovis("FillRecButtons", "myDB", "Customers", rec, FieldList, ¬
                  HCList, False) into rec
end DemoScript
```

DESCRIPTION

Using the hilite of card or bkgnd buttons, this command puts 'Ture' or 'False' into designated fields of a Jovis record.

COMMENTS

This command matches the Jovis fields with card or bkgnd buttons by their relative position in the field list (i.e., item 1 of buttons list to item 1 of record fields list).

If any of the Jovis fields do not exist, an error message is issued and the command is not execut-ed.

If any of the buttons do not exist, a warning is issued and the field value is set for as many Jovis fields as possible

SEE ALSO

FillHCButtons

FillRecFields

SYNTAX

```
Put Jovis (
[1] "FillRecFields", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Record Field List", ¬
[6] "Shell Field List", ¬
[7] '[Card/Bkgnd flag]' )
```

WHERE

[1] is the Jovis 'FillRecFields' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the record used as the source for the card or bkgnd fields;
[5] is a comma delimited list of record fields;
[6] is a comma delmited list of card or bkgnd fields used as the destination;
[7] is an optional flag indicating whether the fields are in the card or bkgnd layer. If this parameter is "false" or missing, the bkgnd layer is used.

EXAMPLE

```
on DemoScript
  put "last_name,first_name,address1" into RecFldList
  Put "Last name,first name,address" into Applist
  put Jovis("FillRecFields", "myDB", "Customers",rec, AdList, ¬
                        RecFldList,False) into rec
end DemoScript
```

DESCRIPTION

Takes the data from the shell application's fields, and puts it into the designated Jovis record fields.

COMMENTS

The command matches the database fields with the shell application's fields by their relative position in the list (i.e., item 1 of the shell application's field list to item 1 of the Jovis record field list).

If any of the database fields do not exist, an error message is issued and the command is not executed. (The original record is returned with no changes.)

If any of the shell application's fields do not exist, a warning is issued and as many fields as possible are filled in the record.

SEE ALSO

FillHCFields

GetCollectionName

SYNTAX

```
      Put Jovis (
      [1] "GetCollectionName", ¬
      [2] "FileGlobal" )
```

WHERE

[1] is the Jovis 'GetCollectionName' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
      on DemoScript
        put Jovis("GetCollectionName","myDB") into FilePath
      end DemoScript
```

DESCRIPTION

Returns the complete path name of the collection opened with given file global.

COMMENTS

By saving off the collection's path name, you can use it for subsequent OpenCollection calls.

SEE ALSO

OpenCollection

GetRecordCriteria

SYNTAX

```
Put Jovis (
[1] "GetRecordCriteria", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Record Path Name]' )
```

WHERE

[1] is the Jovis 'GetRecordCriteria' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional paramter for a record path name.

EXAMPLE

```
on DemoScript
  put Jovis("getRecordCriteria","myDB","Customers","Names") into
         RecCriteria
end DemoScript
```

DESCRIPTION

Returns the search criteria associated with the given record path name.

COMMENTS

This does not replace the current search criteria.

If Record Path Name is not given, this command returns the current search criteria string.

You do not need this command to restore criteria when you restore a record path, because the command 'RestoreRecordPath' automatically restores the correct criteria.

This command is useful for documentation purposes, or to remind the user about which search criteria is associated with which record path name.

SEE ALSO

Record Paths
ClearCriteria
CurrentRecord

GetRecordField

SYNTAX

```
Put Jovis (
[1] "GetRecordField", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Field Name" )
```

WHERE

[1] is the Jovis 'GetRecordField' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the Jovis record to work with;
[5] is the field within the given record to be retrieved.

EXAMPLE

```
on DemoScript
  put  Jovis("getRecordField", "myDB", "Customers", CustomerRec, ¬
                "Account_Start") into AccountDate
end DemoScript
```

DESCRIPTION

Returns the value for the field indicated, from the record. NOTE: Always use this command to get data from a record. You should NEVER access Jovis record fields directly from your scripts. The Jovis record format is not guaranteed to be the same between versions of the database.

COMMENTS

The resulting value may be empty, which is not an error if no value was ever written into the field.

Be sure to check that an error has not occurred.

SEE ALSO

SetRecordField

GetSelectionCriteria

SYNTAX

```
Get Jovis (
[1] "GetSelectionCriteria", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Selection Name]' )
```

WHERE

[1] is the Jovis 'GetSelectionCriteria' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional parameter for a selection name.

EXAMPLE

```
on DemoScript
  put Jovis("GetSelectionCriteria","myDB","Customers","Zips") into
         SelectionCriteria
end DemoScript
```

DESCRIPTION

Returns the criteria associated with the current selection, or if provided, a selection name.

COMMENTS

This does not replace the criteria for the active selection.

If 'Selection Name' is not given, this command returns the criteria for the active selection.

This command will be useful for documentation, or to remind the user about which criteria is associated with which selection name.

If a selection was created with a combination of 'SetSelection', 'AppendSelection', and 'TrimSelection' commands, the selection criteria will be a string containing all the Boolean expressions used in creating the selection.

An empty criteria is: "Field!RecID > [0]"

SEE ALSO

ReserveSelection
SetSelection

GetSelectionField

SYNTAX

```
put Jovis (
[1] "GetSelectionField", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field Name", ¬
[5] "Record #", ¬
[6] '[Named Selection]' )
```

WHERE

[1] is the Jovis 'GetSelectionField' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the field is be accessed;
[5] is the row number within the selection table to be accessed. The first record is Record # 1;
[6] is an optional parameter for a named selection. If this parameter is empty or missing, the active selection is used, assuming it exists.

EXAMPLE

```
on DemoScript
  put Jovis("GetSelectionField","myDB","Customers","First Name",5) into
         firstName
end DemoScript
```

DESCRIPTION

Returns the value of the field indicated from the active selection table for the relation indicated.

COMMENTS

You can not set a selection field, use 'GetSelectionRecord' if you want to change the field's data and then save it to the database using UpdateRecord.

SEE ALSO

GetSelectionRecord

GetSelectionRecord

SYNTAX

```
put Jovis (
[1] "GetSelectionRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record #" )
```

WHERE

[1] is the Jovis 'GetSelectionRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the Record number or row in the selection table to work with. The first record is Record number 1.

EXAMPLE

```
on DemoScript
  put Jovis("GetSelectionRecord","myDB","Customers",5)  into theRec
end DemoScript
```

DESCRIPTION

Returns the complete record from the database that corresponds to the selection record number requested.

COMMENTS

In the multi-user version you must call 'BeginTransaction' if you are going to update this record.

SEE ALSO

SetSelection
CountSelection

GetSelectionStats

SYNTAX

```
Put Jovis (
[1] "GetSelectionStats", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field Name", ¬
[5] '[Named Selection]' )
```

WHERE

[1] is the Jovis 'GetSelectionStats' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the field;
[5] is an optional parameter for a named selection. If this parameter is empty or missing, the active selection is used.

EXAMPLE

```
on DemoScript
  put Jovis("GetSelectionStats","myDB","Customers", "PurchaseItem1") into
         theStats
end DemoScript
```

DESCRIPTION

Returns statistics for the active or named selection, for a field that is either a number or a date.

COMMENTS

The following information is returned:

First line: Minimum value in the field;
Second line: The number of the row in the selection where that value occurs;
Third line: Maximum value in the field;
Fourth line: The number of the row in the selection where that value occurs;
Fifth line: The total (for number fields only);
Sixth line: Average value (for number fields only).

SEE ALSO

SetSelection

GlobalsToRecord

SYNTAX

```
Get Jovis (
[1] "GlobalsToRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Record Field List", ¬
[6] "Shell Globals List" )
```

WHERE

[1] is the Jovis 'GlobalsToRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the Jovis record to use;
[5] is the record's field list to use;
[6] is the shell application's list of globals to use.

EXAMPLE

```
on DemoScript
  put "First_Name,Last_Name,Customer_ID" into fldList
  put Jovis("GlobalsToRecord","myDB","Customers", aRecord,fldList,fldList)
         into aRecord
end DemoScript
```

DESCRIPTION

Takes the data from the shell application's globals, and puts them into the designated Jovis record fields.

COMMENTS

The command matches the database fields with the shell application's fields by the relative position in the list (i.e., item 1 of the shell application's field list to item 1 of the Jovis record field list).

If any of the database fields do not exist, an error message is issued and the command is not executed. (The original record is returned with no changes.)

If any of the shell application's globals do not exist, a warning is issued and as many fields as possible are filled in the record.

ImportData

SYNTAX

```
Get Jovis (
[1] "ImportData", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field List", ¬
[5] "Field Delim", ¬
[6] "Record Delim", ¬
[7] '[File Path]', ¬
[8] '[SFGetPrompt]', ¬
[9] '[Thermometer Prompt]' )
```

WHERE

[1] is the Jovis 'ImportData' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is a comma delimited list of field names for the designated import file. This list should be in the order that the fields occur in the text file;
[5] is the field delimiter used by the import file;
[6] is the record delimiter used by the import file;
[7] is the entire file path for import file. If it is empty, the standard file get dialog is displayed allowing the user to select an import text file;
[8] is the prompt used by the standard file get dialog;
[9] is the prompt used by the thermometer progress dialog.

EXAMPLE

```
on DemoScript
  Put "LastName, Address1, Address2","city,state,zip" into fList
  get Jovis("ImportData", "myDB","Customers",fList)
end DemoScript
```

DESCRIPTION

Imports a text file into a relational database. Never work with an original data file, always import to a copy.

COMMENTS

While the file is being imported, a dialog box is displayed showing the progress. The import process can be stopped by clicking on the cancel button. However, any records imported up to that point will already have been stored in the database. Some or all of these record will have been also indexed. It is very likely, that the database is not stable or useable if the import process is stopped. You should always work from a copy of the data file you are importing to.

SEE ALSO

ExportData
SetProperty

IsTransactionOn

SYNTAX

```
Get Jovis (
[1] "IsTransactionOn", ¬
[2] "FileGlobal" )
```

WHERE

[1] is the Jovis 'IsTransactionOn' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
on DemoScript
  put Jovis("IsTransactionOn", "myDB") into pResult
end DemoScript
```

DESCRIPTION

Returns "True" is the transaction flag at the server is on for the given file. Otherwise, it returns "False".

COMMENTS

This command is non-functional in the single-user, and always returns "false".

SEE ALSO

Transaction Commands

LastRecord

SYNTAX

```
Put Jovis (
[1] "LastRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Criteria]', ¬
[5] '[Function Name]', ¬
[6] '[RecID only flag]' )
```

WHERE

[1] is the Jovis 'LastRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional criteria to be used;
[5] is an optional parameter for function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[6] is an optional 'true' or 'false' parameter to return only the record's ID. If it is 'false' or missing, the whole record is returned, as usual.

EXAMPLE

```
on DemoScript
  put "Field InvoiceNum = " & quote & "0" & quote into criteria
  Put Jovis("LastRecord","myDB","Customers",criteria) into curOrder
end DemoScript
```

DESCRIPTION

Returns the last record in the indicated relation that satisfies the search criteria.

COMMENTS

The search criteria specified will remain in effect until one of the following commands is executed: 'ReadRecord', or 'LastRecord', or 'ClearCriteria'. In the client/server version, the 'RecID' option is useful in conjunction with the delByRecID command, that allows you to delete locked records by passing the !RecID value instead of the entire record.

SEE ALSO

Custom Function Handlers
DelByRecID
PriorRecord
NextRecord

ListFields

SYNTAX

```
Put Jovis (
[1] "ListFields", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] ["FieldsOnly"] )
```

WHERE

[1] is the Jovis 'ListFields' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;

EXAMPLE

```
on DemoScript
  put Jovis("ListFields","myDB","Customers") into fld "fieldNames"
end DemoScript
```

DESCRIPTION

Returns a list of all the fields and their types for the given relation. If the optional fourth parameter is "FieldsOnly", the field names will returned in a comma delimited list.

SEE ALSO

ListIndexes

ListIndexes

SYNTAX

```
Put Jovis (
[1] "ListIndexes", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'ListIndexes' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;

EXAMPLE

```
on DemoScript
  put Jovis("ListIndexes","myDB","Customers") into order_Index_List
end DemoScript
```

DESCRIPTION

Returns a list of all the indexes for the given relation

COMMENTS

If an index is invalid or deleted, 'ListIndexes' will indicate this by putting the word “Invalid” after the index name.

SEE ALSO

Invalid index

ListRecordPaths

SYNTAX

```
Put Jovis (
[1] "ListRecordPaths", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'ListRecordPaths' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;

EXAMPLE

```
on DemoScript
  put Jovis("ListRecordPaths","myDB","addressBook") into RecordPathList
end DemoScript
```

DESCRIPTION

Returns a list of record path names for the given relation.

COMMENTS

The current record path, if there is one, is not included in this list.

SEE ALSO

ReserveRecordPath
Record Paths

ListRelations

SYNTAX

```
Put Jovis (
[1] "ListRelations", ¬
[2] "FileGlobal" )
```

WHERE

[1] is the Jovis 'ListRelations' command;
[2] is the global identifier for the file you want to work with.

EXAMPLE

```
on DemoScript
  put Jovis("ListRelations","myDB") into cd fld "relationNames"
end DemoScript
```

DESCRIPTION

Returns a list of relation names for the given database.

COMMENTS

If a relation is invalid, 'ListRelations' indicates this by putting the word “Invalid” after the relation name.

SEE ALSO

Invalid Relation

ListSelectionFields

SYNTAX

```
Put Jovis (
[1] "ListSelectionFields", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Named Selection]' )
```

WHERE

[1] is the Jovis 'ListSelectionFields' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional parameter for a named selection.

EXAMPLE

```
on DemoScript
  put Jovis("ListSelectionFields","myDB","Customers") into SFList
end DemoScript
```

DESCRIPTION

Returns a list of the fields in the current or named selection.

COMMENTS

Returns a list of the fields in the active selection.

SEE ALSO

SetSelection
GetSelectionField

ListSelections

SYNTAX

```
Put Jovis (
[1] "ListSelections", ¬
[2] "FileGlobal", ¬
[3] "RelationName" )
```

WHERE

[1] is the Jovis 'ListSelections' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;

EXAMPLE

```
on DemoScript
  put Jovis("ListSelections","myDB","Customers") into SelectionsList
end DemoScript
```

DESCRIPTION

Returns a list of named selections for the given relation.

SEE ALSO

ReserveSelection

Merge

SYNTAX

```
Get Jovis(
[1]   "Merge", ¬
[2]   "FileGlo,RelationName,SelName or ["Current"],CompareFldName", ¬
[3]   "Left Table Field List", ¬
[4]   "FileGlo,RelationName,SelName or ["UseRelation" or "Current"]
            ,CompareFldName",[n] ¬
[5]   "Right Table Field List", ¬
[6]   "FileGlo,RelationName,NewSelName", ¬
[7]   ['MergeType'"inclusive", "exclusive"] ¬
[8]   ['Progress prompt'] )
```

WHERE

[1] is the Jovis 'Merge' command;
[2] a four item quoted string consisting of a Jovis global name, a relation name, either a named selection or the literal word "Current", and finally the name of a compare field
[3] an itemized list of fields available for use in named selection in parameter [2]
[4] a four or five item quoted string consisting of a Jovis global name; a relation name; either a second named selection or the literal word "Current" or "UseRelation"; the name of the compare field; a fifth item is to be used if the compare field is of type 'text', it indicates the number of characters up to 32 to compare. The max is 32 and is also the default.
[5] an itemized list of fields available for use in the named selection in parameter [4]
[6] a three item quoted string consisting of a Jovis global name, a relation name, and the name to use for the newly merged selection
[7] the merge type, either "Inclusive" or "Exclusive". This parameter is optional, and can be left empty. The default, is "Exclusive".
[8] if the merge operation takes more than 2 seconds, a thermometer dialog will appear. Use this parameter for the prompt displayed for this dialog. (To suppress the dialog's appearance, use the SetProperty command's "delay" property to zero.) This parameter is optional, the defalut message is: "Merging data using table: [relation name]".

EXAMPLE

```
on DemoScript
  put "" into cd fld "Display"
  -- delete previously named selection
  if "MERGED_TABLE" is in Jovis("ListSelections","myDB","Customers") then
    get Jovis("delSelection","myDB","Customers","MERGED_TABLE")
  end if
  --
  put "Field Account_Start ≥ [1/1/95]" into aCriteria
  put "First_Name,Last_Name,Customer_ID" into fldList
  get Jovis("SetSelection","myDB","Customers",fldList,aCriteria)
  --
  get Jovis( "Merge",¬
  "myDB,Customers,Current,Customer_ID",¬
  "First_Name,Last_Name",¬
  "myDB,Purchases,UseRelation,Customer_ID",¬
  "Item_Descrip,Customer_ID",¬
  "myDB,Customers,MERGED_TABLE","Exclusive" )
  --
  put Jovis("ListSelections","myDB","Customers") into temp
  --
  put "First_Name,Last_Name,◊,Item_Descrip,Customer_ID" into fldList
  put Jovis("SelectionToVar","myDB","Customers",comma,¬
  return,fldList,"","MERGED_TABLE") into cd fld "Display"
end DemoScript
```

DESCRIPTION

The Merge command provides the ability to create selections across multiple relations. This capability is commonly know as a 'Join' in the SQL language. You create a "Merged Selection" in two steps: First, create a selection using the SetSelection command, and second, use the Merge command to generate a new named selection using a compare field. The compare field must be present for determining the merged selection. This two step process makes it possible to create merged selections with great ease and tremendous flexibility.

COMMENTS

See the special "Merge" tutorial in this manual for additional information.

SEE ALSO

SetProperty
Multiple Selections Tutorial

NextRecord

SYNTAX

```
      Put Jovis (
      [1] "NextRecord", ¬
      [2] "FileGlobal", ¬
      [3] "RelationName", ¬
      [4] '[Function Name]', ¬
      [5] '[RecID Only flag]' )
```

WHERE

[1] is the Jovis 'NextRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional parameter for the function that you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[5] is an optional 'true' or 'false' parameter to return only the record's ID. If it is 'false', empty or missing, the whole record is returned, as usual.

EXAMPLE

```
      on DemoScript
        put Jovis("NextRecord","myDB","Customers") into aRecord
      end DemoScript
```

DESCRIPTION

Returns the next record in the indicated relation which satisfies the search criteria used by the ReadRecord command.

COMMENTS

In the client/server version the 'RecID' option is useful in conjunction with the delByRecID command, that allows you to delete locked records by passing the !RecID value instead of the entire record.

SEE ALSO

ReadRecord
Custom Function Handlers

OpenCollection

SYNTAX

```
Get Jovis (
[1] "OpenCollection", ¬
[2] "FileGlobal", ¬
[3] '[File Pathname]', ¬
[4] '[SFGet Dialog Prompt]', ¬
[5] '[Read Only]', ¬
[6] '[Write Only Password]', ¬
[7] '[Exclusive flag]', ¬
[8] '[SFGet Dialog Coors.]', ¬
[9] '[Type[,Creator]]' )
```

WHERE

[1] is the Jovis 'OpenCollection' command;
[2] is the global identifier for the file you want to work with;
[3] is the entire path name to the data file. If it is empty, the standard file get dialog box will be displayed;
[4] is the prompt for the standard file get dialog if it is used;
[5] if "True", the data file is set to 'Read-Only';
[6] is the write only password
[7] is recognized only in the multi-user version (See the Exclusive status section). In the single-user version, it is ignored.
[8] is the standard file get dialog box coordinates.

EXAMPLES

```
on DemoScript1
  global myDB
  if myDB is empty then
    put "Jovis" into myDB
  else
    answer "Invalid file global"
    exit DemoScript1
  end if
  get Jovis("OpenCollection", "myDB")
end DemoScript1
```

```
on DemoScript2
  global myDB
  if myDB is empty then
    put "Jovis" into myDB
  else
    answer "Invalid file global"
    exit DemoScript2
  end if
  put "myWriteOnlyPW" into myPassword
  get Jovis("OpenCollection","myDB","","Which data file do you
       want?",true,myPassword)
end DemoScript2

on DemoScript3
  global myGlobalVariable
  if myGlobalVariable is empty then
    put "Jovis" into myGlobalVariable
  else
    answer "Invalid file global"
    exit DemoScript3
  end if
  -- Client/Server
  get Jovis("OpenCollection "myGlobalVariable","my datafile:myserver@*")
end DemoScript3
```

DESCRIPTION

Opens an existing relational database file. A global variable must be initialized to “Jovis” before using this command. (For example: put "Jovis" into myGlobal.) OpenCollection changes this global to contain data about the opened data file.

COMMENTS

Always check the JovisErrorCode global before proceeding further.

The correct password (if one has been assigned to the file) is required before any file can be opened. If a password has been assigned to the data file, and is not given as a parameter, an error message is returned and the file is NOT opened.

SEE ALSO

GetCollectionName
Exclusive Status
Error and Warning Handlers
Initializing a Jovis Global

PriorRecord

SYNTAX

```
Put Jovis (
[1] "PriorRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Function Name]', ¬
[5] '[RecID Only flag]' )
```

WHERE

[1] is the Jovis 'PriorRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional parameter for a function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[5] is an optional 'true' or 'false' parameter to return only the record's ID. If it is 'false', empty or missing, the whole record is returned, as usual.

EXAMPLE

```
on DemoScript
  put Jovis("PriorRecord","myDB","Customers") into curOrder
end DemoScript
```

DESCRIPTION

Returns the prior record in the indicated relation which satisfies the previously specified search criteria.

COMMENTS

In the client/server version, the "!RecID" option is useful in conjunction with the delByRecID command, that allows you to delete locked records by passing the !RecID value instead of the entire record.

SEE ALSO

ReadRecord
Custom Function Handlers
LastRecord

ReadRecord

SYNTAX

```
Put Jovis (
[1] "ReadRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Criteria]', ¬
[5] '[Function Name]', ¬
[6] '[RecID Only flag'] )
```

WHERE

[1] is the Jovis 'ReadRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is an optional criteria to be used; the default is: "Field!RecID > [0] ".
[5] is an optional parameter for a function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[6] is an optional 'true' or 'false' parameter to return only the record's ID. If it is 'false', empty or missing, the whole record is returned, as usual.

EXAMPLE

```
on DemoScript
  put "Field Customer ID = " & quote& "4432" & quote into criteria
  Put Jovis("ReadRecord","myDB","Customers",criteria)  into aRecord
end DemoScript
```

DESCRIPTION

Returns the first record in the indicated relation that satisfies the search criteria.

COMMENTS

The search criteria specified will remain in effect until one of the following commands is executed: 'ReadRecord', or 'LastRecord', or 'ClearCriteria'. In the client/server version the 'RecID' option is useful in conjunction with the delByRecID command, that allows you to delete locked records by passing the !RecID value instead of the entire record.

SEE ALSO

NextRecord
Custom Function Handlers
LastRecord
PriorRecord

RecordToGlobals

SYNTAX

```
Get Jovis (
[1] "RecordToGlobals", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Record Field List", ¬
[6] "Globals List" )
```

WHERE

[1] is the Jovis 'RecordToGlobals' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the Jovis record to use;
[5] is a comma delimiter list of record fields to use;
[6] is a comma delimiter list of names of globals to use.

EXAMPLE

```
on DemoScript
  get Jovis("RecordToGlobals", "myDB", "Customers", "Record", ¬
                 "RecFldList", "GlobalsList")
end DemoScript
```

DESCRIPTION

Takes data from a Jovis record and populates the shell application's globals with the field data from the record.

COMMENTS

This command matches the database fields with the shell application's globals by their relative position in the list (i.e., item 1 of record field list to item 1 of the global's list).
If any of the Jovis fields do not exist, an error message is issued and the command is not executed.
If any of the globals do not exist, they will be automatically created.

SEE ALSO

GlobalsToRecord

RecordToVar

SYNTAX

```
Put Jovis (
[1] "RecordToVar", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Record Field List"
[6] "Delimiter" ) into Result
```

WHERE

[1] is the Jovis 'RecordToGlobals' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the Jovis record to use;
[5] is a comma delimiter list of record fields to use.
[6] is the character used for delimiting the record fields.

EXAMPLE

```
on DemoScript
  put Jovis("RecordToVar","myDB","Customers","Record","RecFldList",comma)
         into jResult
end DemoScript
```

DESCRIPTION

Concatenates the record fields listed in parameter five. Each field is delimited with the character provided in parameter six.

COMMENTS

This command is annlogous to the 'SelectionToVar' command. Never access record fields directly from your scripts. You should use either this commend or, 'GetRecordField' for retrieving a record's field information. The record format is not guaranteed to be remain the same in future versions.

SEE ALSO

SelectionToVar

RenameField

SYNTAX

```
Get Jovis (
[1] "RenameField", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Old Field Name", ¬
[5] "New Field Name" )
```

WHERE

[1] is the Jovis 'RenameField' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the field whose name you want to change;
[5] is the name you want to begin using.

EXAMPLE

```
on DemoScript
  get Jovis("RenameField",s "myDB","Customers","zip","long_zip")
end DemoScript
```

DESCRIPTION

Changes the name of the indicated field to the new name.

COMMENTS

The new field name must follow the standard rules for naming Jovis fields. Note that the field type does not change. If a field was indexed, the name of the index is changed automatically. The actual index remains as before. In the client/server version the exclusive flag must set to true.

SEE ALSO

CreateField
CreateIndex

ReserveRecordPath

SYNTAX

```
Get Jovis (
[1] "ReserveRecordPath", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record Path Name" )
```

WHERE

[1] is the Jovis 'ReserveRecordPath' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name for the reserved record path.

EXAMPLE

```
on DemoScript
  get Jovis("ReserveRecordPath", "myDB","Customers","Names")
end DemoScript
```

DESCRIPTION

Removes the active record path and reserves it in memory, identified by the given name.

COMMENTS

When this command finishes, the current active record path is cleared. In other words, there is no current or active record path.

SEE ALSO

RestoreRecordPath
Record Paths

ReserveSelection

SYNTAX

```
Get Jovis (
[1] "ReserveSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Selection Name" )
```

WHERE

[1] is the Jovis 'ReserveSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name for the reserved selection.

EXAMPLE

```
on DemoScript
  get Jovis("ReserveSelection", "myDB","Customers","Zipcodes")
end DemoScript
```

DESCRIPTION

Removes the active selection and reserves it in memory, identified by the given name.

COMMENTS

When this command finishes, the active selection is cleared; there is no current or active selection.

SEE ALSO

RestoreSelection
Multiple Selections

RestoreRecordPath

SYNTAX

```
Get Jovis (
[1] "RestoreRecordPath", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record Path Name" )
```

WHERE

[1] is the Jovis 'RestoreRecordPath' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the record path to be resored.

EXAMPLE

```
on DemoScript
  get Jovis("RestoreRecordPath", "myDB","Customers","Names")
end DemoScript
```

DESCRIPTION

Removes the current record path and replaces it with the record path named by the fourth parameter. The named record path becomes the active record path.

SEE ALSO

Record Paths
ReservedRecordPath
DupRecordPath

RestoreSelection

SYNTAX

```
Get Jovis (
[1] "RestoreSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Selection Name" )
```

WHERE

[1] is the Jovis 'RestoreSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the named selection to be restored.

EXAMPLE

```
on DemoScript
  get Jovis("RestoreSelection", "myDB","Customers","Zips")
end DemoScript
```

DESCRIPTION

Removes the current selection and replaces it with the selection named by the fourth parameter. The named selection becomes the active selection.

SEE ALSO

Multiple Selections
ReserveSelection

SaveFile

SYNTAX

```
Get Jovis (
[1] "SaveFile", ¬
[2] "FileGlobal", ¬
[3] '[All]' )
```

WHERE

[1] is the Jovis 'SaveFile' command;
[2] is the global identifier for the file you want to work with;
[3] is an option to save all open database files.

EXAMPLE

```
on DemoScript
  get Jovis ("SaveFile", "myDB")
end DemoScript
```

DESCRIPTION

Flushes all changes made to the data file to disk. See the 'AutoSave' property for additional information.

COMMENTS

With the "SaveFile" command you have everything necessary to guarantee that the database is saved and that changes are secure.

SEE ALSO

SetProperty

SelectionToVar

SYNTAX

```
Put Jovis (
[1] "SelectionToVar", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] '[Field Delim]', ¬
[5] '[Record Delim]', ¬
[6] '[Field List]', ¬
[7] '[Record Range]', ¬
[8] '[Named Selection]' )
```

WHERE

[1] is the Jovis 'SelectionToVar' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the field delimiter to use. The default is the tab character;
[5] is the the record delimiter to use. The default is the carriage return character;
[6] is a comma delimited list of record fields to be used;
[7] If you do not wish to use all the records in the selection, you may specify a range of record numbers instead. If you give one number for the parameter, then the record range will be from the first selection record up to the number you have given. If you give two numbers, separated by a comma, then the record range will be all the records in that range, including the two given as parameters. If the parameter is empty or missing, then all the records in the selection are used.
[8] is the name of a selection to use instead of the current one.

EXAMPLE

```
on DemoScript
  put Jovis("SelectionToVar","myDB","Customers") into field "Order_List"
  --
  Put "first_name,last_name,telephone" into Names_List
  put Jovis("SelectionToVar","myDB","Customers","","",Names_List,"25")
         into field "Order List"
  --
  Put Jovis("SelectionToVar","myDB","Customers","","",Names_List,"50,100")
         into ReportVar
end DemoScript
```

DESCRIPTION

Exports the active or named selection to a script variable.

COMMENTS

With Field List, you also give the order in which the fields appear in the variable. This works equally well with all the fields, or a subset of the fields in the selection.

SEE ALSO

SetSelection
SortSelection
RestoreSelection

SetProperty

SYNTAX

```
Get Jovis (
[1] "SetProperty", ¬
[2] "FileGlobal", ¬
[3] "Property Name", ¬
[4] "Value" )
```

WHERE

[1] is the Jovis 'SetProperty' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the property you want to set;
[4] is the value that the property is being set to.

EXAMPLE

```
on DemoScript
  get Jovis("SetProperty", "myDB","Delay","0")
end DemoScript
```

DESCRIPTION

Delay:	If the value of the property is zero, Jovis will not initialize or display either the Progress or Thermometer windows. Once displayed, the updating is once every 2 seconds.
ErrorGlobal:	It is possible to change the name of the error global. The default is "JovisErrorCode". You can change this global at any time.
ErrorMsg:	The name of the Jovis error message. The default is "JovisErrorMsg". A complete description of it's use can be found under "Error and Warning Handlers".
WarningMsg:	The name for the Jovis warning message. The default is "JovisWarningMsg". A complete description of it's use can be found under "Error and Warning Handlers".
AutoSave:	By default, auto saving is turned on when you open a file. Setting "AutoSave" to "true" turns auto saving on, and "false" turns it off. You can turn it on or off at anytime from your scripts. If you set this property to "false", you can

	manually perform file saving with the 'SaveFile' command. Generally, this property shoud always be set to "True". Exceptions might include, batch pro-coessing of record updates, or when using the 'ImportData' command.
Loc:	Global coordinates for the 'Progress' and 'Thermometer' dialog boxes.
ShellType:	With the advent of standalone applications, it's no longer a simple task for Jovis to know what type or kind of environment it is installed in. By setting this property to either "HyperCard", "SuperCard", "OMO", or "Director", Jovis can better handle error messaging and other differences between the various "Shell" applications.
WindowName:	Name of the 'Progress' and 'Thermometer' windows.

COMMENTS

The 'SetProperty' command applies to an individual open data file, not ALL currently open files.

SEE ALSO

SaveFile

SetRecordField

SYNTAX

```
Put Jovis (
[1] "SetRecordField", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record", ¬
[5] "Field Name", ¬
[6] "Value" )
```

WHERE

[1] is the Jovis 'SetRecordField' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the record whose field is being set;
[5] is the name of the field to set;
[6] is the new information to be new information to be installed into the record's idicated field.

EXAMPLE

```
on DemoScript
  put Jovis("SetRecordField","myDB","Customers",CurRec,"first_name","Joe")
         into CurRec
end DemoScript
```

DESCRIPTION

Puts the value passed as parameter [6] into the indicated field in the record.

Never access a record's field information directly from your scripts. Always use GetRecordField and SetRecordField for these operations. The record's format may change in the future.

COMMENTS

Always check for errors. If an error is encounter, the record will be returned unaltered.

SEE ALSO

GetSelectionRecord
UpdateRecord
Transaction Commands

SetSelection

SYNTAX

```
Get Jovis (
[1] "SetSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field List", ¬
[5] '[Criteria]', ¬
[6] '[Function Name]', ¬
[7] '[Prompt]', ¬
[8] '[Page Size]', ¬
[9] '[Continue flag]' )
```

WHERE

[1] is the Jovis 'SetSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is a comma delimited list of field names. Only those fields listed in this list are included in the selection. If the field list is a "*", all fields are included.
[5] is the criteria for creating the selection. The default is "Field !RecID > [0]".
[6] is an optional parameter for a function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers";
[7] is a prompt used by the progress dialog.
[8] indicates the max number of rows for a created selection;
[9] if "false" or missing, the given criteria replaces any previous criteria. If this flag is "true", the 'current' criteria is used, and further selecting continues. Note: a previous call with this parameter set to "true" replaces all records previously selected thereby producing a page-in page-out result. See "AppendSelection" for appending a current selection with the continue flag.

EXAMPLE

```
on DemoScript
  ask "Enter last name to set selection to:"
  if the result = "cancel" then exit DemoScript
  put it into Last_Name
  Put "Field Last_Name = [" & Last_Name &"]" into into criteria
  Put "Last_Name,phone,Customer_ID,Account_Start" into fieldList
  get Jovis("SetSelection", "myDB", "Customers", fieldList, Criteria)
end DemoScript
```

DESCRIPTION

Creates a selection table for the relation indicated. This selection is called the “active” or "current" selection.

COMMENTS

Criteria is used to restrict which records are selected. If the Criteria parameter is empty, all records in the relation are selected. If criteria is invalid, no new selection is created, and the 'active' or 'current' selection is not replaced.

SEE ALSO

Page Selecting
Selection Criteria
AppendSelection
SelectionToVar
SortSelection

ShutDown

SYNTAX

```
Get Jovis (
[1] "ShutDown" )
```

WHERE

[1] is the Jovis 'ShutDown' command;

EXAMPLE

```
on DemoScript
  get Jovis("ShutDown")
end DemoScript
```

DESCRIPTION

This command is required, and MUST be called after your final Jovis call has been made. (Usually, a 'CloseCollection' or 'dbClose' precedes this command.) Just as with, "StartUp" it calls directly into the Jovis system. Before deallocating memory used by cache memory system, it will make sure everything that needs writing to disk is written out.

COMMENTS

THIS COMMAND IS REQUIRED!

SEE ALSO

StartUp

SortSelection

SYNTAX

```
Get Jovis (
[1] "SortSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Field Name [,A or D]", ¬
[5] '[Field Name [,A or D]', )
```

WHERE

[1] is the Jovis 'SortSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the name of the primary field to sort either ascending or descending against;
[5] is the secondary field name to sort, etc.

EXAMPLE

```
on DemoScript
  get Jovis("SortSelection", "myDB","Customers","OrderDate,d","orderNo")
end DemoScript
```

DESCRIPTION

Sorts the selection table for the relation indicated. You can sort on up to 12 different fields.

COMMENTS

A key parameter consists of a field name followed by a comma and the character “A” or “D” (for Ascending or Descending). Key1 is the primary sort key, Key2 is the secondary, and so on up to a maximum of 12 sort keys. The SortSelection command does not use the progress or thermometer dialogs. For particularly large selections you should at least set the 'watch' cursor.

SEE ALSO

SetSelection

StartUp

SYNTAX

```
      Get Jovis (
      [1] "StartUp", ¬
      [2] '[Cache Size]' )
```

WHERE

[1] is the Jovis 'StartUp' command;
[2] is the size of the cache system to be allocated.

EXAMPLE

```
      on DemoScript
        get Jovis("StartUp", 512000*2)  --> 1Meg.
      end DemoScript
```

DESCRIPTION

The StartUp command must be called BEFORE any other Jovis calls. It initializes the cache memory system as well as other system managers. It requires one parameter which is the size of the memory allocation for the cache memory system. By default the size is 512K. If you pass anything less, Jovis uses the default size. The "Startup" command is optional so that you can call 'CreateCollection', 'OpenCollection', 'dbInit', or 'dbOpen' and still initialize the Jovis cache system, but only to the default size. If there are any errors during initialization, they will be reported to you via the JovisErrorCode global, not with the error function handler.

COMMENTS

Used only by the single-user version. If called by the multi-user version, nothing will occur.

SEE ALSO

Shutdown

TrimSelection

SYNTAX

```
Get Jovis (
[1] "TrimSelection", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Criteria", ¬
[5] '[Function Name]' )
```

WHERE

[1] is the Jovis 'TrimSelection' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the criteria used for trimming the selection;
[5] is an optional parameter for function which you provide in your scripts. It must be located at the card level or higher. See "Custom Function Handlers".

EXAMPLE

```
on DemoScript
  Put "Field zip = " & quote & "60612" & quote into theCriteria
  get Jovis("TrimSelection", "myDB","Customers",theCriteria)
end DemoScript
```

DESCRIPTION

Compares the records in the active selection and removes those not matching the new criteria.

COMMENTS

You can not use TrimSelection on a merged selection.

SEE ALSO

SetSelection
AppendSelection
SortSelection
SelectionToVar
Custom Function Handlers
Merge

UpdateRecord

SYNTAX

```
Get Jovis (
[1] "UpdateRecord", ¬
[2] "FileGlobal", ¬
[3] "RelationName", ¬
[4] "Record" )
```

WHERE

[1] is the Jovis 'UpdateRecord' command;
[2] is the global identifier for the file you want to work with;
[3] is the name of the relation you want to work with within the designated file;
[4] is the record to be updated.

EXAMPLE

```
on DemoScript
  get Jovis("UpdateRecord", "myDB","Customers",orderRecord)
end DemoScript
```

DESCRIPTION

Writes changes to the record into the database file.

COMMENTS

1. Until you do an updateRecord, the changes you have made in the record's fields using SetRecordField, are not saved to the database.

2. For the client/server version changes made with updateRecord are stored at the Server and the collection is not physically updated until the CommitTransaction command is executed.

SEE ALSO

SetRecordField
Transaction Commands