Apple Event Registry: Database Suite

Developer Technical Publications

APPLE COMPUTER, INC.

Apple Computer, Inc.
20525 Mariani Avenue
Cupertino, CA 95014-6299
408-996-1010

Simultaneously published in the United States and Canada.

LIMITED WARRANTY ON MEDIA AND REPLACEMENT

If you discover physical defects in the manual or in the media on which a software product is distributed, APDA will replace the media or manual at no charge to you provided you return the item to be replaced with proof of purchase to APDA.

ALL IMPLIED WARRANTIES ON THIS MANUAL, INCLUDING IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED IN DURATION TO NINETY (90) DAYS FROM THE DATE OF THE ORIGINAL RETAIL PURCHASE OF THIS PRODUCT.

Even though Apple has reviewed this manual, **APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS MANUAL, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS MANUAL IS SOLD "AS IS," AND YOU, THE PURCHASER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS MANUAL, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

Figures and tables

The Database Suite

The Database suite defines Apple event constructs that are used for communicating with database programs and transferring data to and from a database. This suite should be supported by applications that work cooperatively with database applications.

Introduction to the Database suite

The Database suite defines Apple event constructs that allow applications to communicate with database programs. By using the constructs defined in the Database suite, an application should be able to access a database and manipulate its data.

The Database suite is an extension of the events and objects defined in the *Apple Event Registry*: Standard Suites. In order to use the Database suite Apple event constructs, an understanding of the events and objects of the Core suite, Table suite, and Miscellaneous Standards is required. Anyone attempting to understand or use the Database suite must read and understand the *Apple Event Registry*: Standard Suites. An understanding of the Apple Event Manager is also encouraged. (See the Apple Event Manager chapter of "*Inside Macintosh: InterApplication Communication*" for more information.)

Overview of the Database suite

The Database suite defines Apple events for

- performing more complex transactions than required by the Core suite. A *transaction* is a series of Apple events that must be processed without interruption from other Apple events. The Core suite transaction model allows a single transaction; any subsequent transactions cannot be started until the initial transaction has been completed. On the other hand, databases typically allow multiple transactions that can consist of a variety of operations. These transactions can be canceled without completion by using the Abort Transaction event, causing the database to return to the state it was in prior to the start of the transaction. Transactions can also be verified through the use of cSession object.
- sorting or accessing tables within a database. Data in a database is stored in the form of tables. A *table* is a named row of column headings, with zero or more rows of data values inserted under those headings. A single record is called a *row* and a table consists of a set of rows. The intersection of a row and a column is called a cell. The data contained in a cell is known as *atomic data* which can have either a single value or a list of values (in the case of multi-valued data cells).
- producing grouped summaries from tables.

The Database suite also defines object classes for

- accessing rows, columns, and cells within a database.
- accessing a database management system (DBMS). A DBMS is a collection of programs that enables users to create and maintain a database.

- accessing the database host (a computer running a DBMS).
- managing the sessions object on a host, DBMS, or a database.
- accessing selections in a database.
- accessing key fields in a database. The *key field* in a database is a column (or a combination of a number of columns) that can be used to uniquely identify rows in the table.

Applications that should support the Database suite

The following types of applications should support the Database suite:

- Applications that manipulate databases or spreadsheets.
- Applications that communicate with local or remote databases.
- Applications that remotely access databases.

Typical client applications for the Database suite

The following types of applications are likely to be clients of applications that support the Database suite:

- Applications that utilize shared data, for example, addresses and phone numbers.
- Applications that use information in a networked database, such as CAD, Forms, Inventory Control, or Point of Sale.
- Applications that interact with database programs.

Relationship of the Database suite with other suites

The Database suite's Apple event constructs allow applications to refer to, request, and modify a remote database application's data when used in conjunction with the objects and events defined in the *Apple Event Registry*: Standard Suites.

The Database suite defines three new Apple events: Abort Transaction, Group, and Sort, in addition to the Begin Transaction event which is an extension of the Begin Transaction event of the Miscellaneous Standards. To support the Database suite, an application must also support a small subset of Apple events from the Core suite, particularly Do Objects Exist, Get Data, Get Data Size, and Set Data. A few Apple

events from the Miscellaneous Standards are also used by the Database suite. All of the common functions of a database operation can be accomplished using only the Apple events defined in the *Apple Event Registry*: Standard Suites.

For example, the Create Element event is used to create an object in this suite. A descriptor record containing initialization data is passed as the keyAEData parameter to the Create Element event. This descriptor contains optional information necessary to create the object. If the information is missing, the server uses default values. It follows that the Get Data and Set Data events are the most commonly used events of the Core suite. They provide the capability of manipulating the data defined by the database objects. By using the Get Data event, an application can interrogate another application about the value of various fields, columns, rows, or cells. The Apple Event Manager's automatic coercion handlers allow the user to request data in a format acceptable to Apple event constructs.

Using object specifiers in place of other parameters

In all of the suites except the Finder suite, you can substitute an object specifier for any parameter of an Apple event that is not already defined as an object specifier. This object specifier must specify a single object. When you substitute an object specifier for a parameter, the actual value of the parameter is the value of the default descriptor record for the specified object (that is, the value you get when you send a Get Data Apple event for the object and do not specify a particular descriptor type for the result).

Apple events defined in the Database suite

The Apple events defined in the Database suite are described in the following sections. Table 1 lists these Apple events.

■ **Table 1** Apple events defined in the Database suite

Name	Requested action
Abort Transaction	Cancel a series of changes.
Begin Transaction	Begin a transaction thread and optionally associate it with an existing session.
Group	Summarize a table.
Sort	Order an object by one or more of its elements.

Abort Transaction—cancel a series of changes

The Abort Transaction Apple event is used to cancel a transaction without allowing it to complete. This ensures that any changes that took place during the transaction are not made permanent.

Event Class kAEDatabase

Event ID kAEAbortTransaction

Parameters None

Reply Parameters None

Notes Abort Transaction complements the Begin Transaction, End Transaction, and Transaction Terminated events found in the Miscellaneous Standards. It extends their functions to allow commit and rollback functionality. Transactions are used to connect a series of events into one logical operation. The transaction ID, which is present in every Apple event, is used to identify the transaction. If a session has been associated with a given transaction, every event that is part of that transaction will behave as if the session were explicitly given. To associate a session with a transaction, include the session identifier as an optional parameter in the Begin Transaction event. Any transaction can either be completed by sending an End Transaction event (known as *commit* because it saves all the pending actions), or it may be aborted by sending an Abort Transaction event (known as *rollback* because it cancels any pending series of changes).

Databases that do not support commit and rollback should treat the Begin Transaction event as a Save and the Abort Transaction event as a Revert. Otherwise, the server returns errAEEventNotHandled on receipt of the Abort Transaction event to indicate that it could not rollback the transaction. More complex database management systems are able to rollback the transaction on receipt of an Abort Transaction event or commit the transaction upon receipt of the End Transaction event.

If a database does not support any type of transaction, the Begin, Abort, and End Transaction events are not handled. Instead, the events are processed on a first come first serve basis, which is an inefficient method for even the least complex database. On the other hand, some databases support only single transaction threads that handle one transaction at a time. In such cases, the Begin Transaction and the End Transaction events are used to lock out other events while processing a series of events. The way these events are handled by the server depends on the way the server has been implemented. Either of these transaction mechanisms may be used with or without support for cSession object.

The Abort Transaction event differs from the Transaction Terminated event found in the Miscellaneous Standards, in that the client application sends an Abort Transaction event to cancel a transaction, whereas in the latter case, the server sends a Transaction Terminated event if it encounters an error.

Result Codes

errAEEventFailed	–10000	The Apple event handler failed when attempting to handle the Apple event.
errAENoSuchTransaction	-10012	The specified transaction is not a valid transaction; the transaction may never have begun, or it may have been terminated.

Begin Transaction—begin a transaction thread and optionally associate it with an existing session

The Begin Transaction Apple event is an extension of the Begin Transaction event of the Miscellaneous Standards. The Begin Transaction Apple event is used to initiate a transaction and return a transaction ID for subsequent events in the transaction. In the Database suite, this Apple event is extended to provide a way to associate a transaction thread (a series of events that occur under the auspices of a single transaction) with an existing session. The cSession object, the direct parameter to the Begin Transaction Apple event, is optional and may not be required even where sessions are being used. The client application may be allowed to open a transaction without any session object. In such cases, the server uses guest privileges for the transaction. The direct parameter is optional as some applications may not require sessions. On servers that do require sessions, this parameter is required for a transaction to be initiated.

Event Class kAEMiscStandards

Event ID kAEBeginTransaction

Parameters

keyDirectObject

Description:	The session with which to associate this transaction
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Optional

Reply Parameters

keyAEResult

Description:	The transaction ID
Descriptor Type:	typeLongInteger
Required or Optional?	Required

keyErrorNumber

Description:	The result code for the event
Descriptor Type:	typeLongInteger
Required or Optional?	Optional (The absence of a keyErrorNumber parameter in the reply indicates that the event was handled successfully.)

keyErrorString

Description:	A character string that describes the error, if any, that occurred when the event was handled
Descriptor Type:	typeIntlText
Required or Optional?	Optional

Result Codes

errAEEventFailed	–10000	The Apple event handler failed when attempting to handle the Apple event.
errAEInTransaction	–10011	Could not handle this Apple event because it is not part of the current transaction.
errAENoSuchTransaction	–10012	The specified transaction is not a valid transaction; the transaction may never have begun, or it may have been terminated.

Notes Some applications may wish to allow nested transactions. A *nested transaction* is a new transaction within the context of another existing transaction. Although nested transactions are allowed, they are not required. Nested transactions allow multiple levels of commit and rollback to occur. If an application does not support nested transactions it may return an errAEInTransaction error if a Begin Transaction event occurs during another transaction. Initial Begin Transaction events use a transaction ID of kAnyTransaction, whereas nested Begin Transaction events use their enclosing transaction's ID.

Group—summarize a table

The Group Apple event is used to create a table of summary rows by using data from an existing table. These summary rows are computed based on the columns specified: for every column in the direct object, all rows are grouped by value, and a summary row is produced for each row with a distinct value. Rows with identical values are summarized by using the function specified on the specified group columns. Functions are listed in the constants section. Either or both of the group columns and functions may be a list. If there is a list of columns and a single function, then the function applies to all the columns. If there are multiple functions, then there must be only one function for every column listed. This makes it possible to summarize multiple columns with one Apple event.

Event Class kAEDatabase

Event ID kAEGroup

Parameters

keyDirectObject

Description:	The columns to use in generating a new table of summary rows
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Required

keyAEGroupColumns

Description:	The columns to summarize (may be a list)
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Required

keyAEGroupFunctions

Description:	The functions with which to summarize (may be a list). The following group functions are available: kAverage, kCount, kMaximum, kMean, kMinimum, kStdDev, and kSum.
Descriptor Type:	typeEnumeration
Required or Optional?	Required

keyAEInsertHere

Description:	The destination summary table
Descriptor Type:	typeInsertionLoc
Required or Optional?	Optional

Reply Parameters

keyAEResult

Description:	The summary table (if not specified)
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Optional

keyErrorNumber

Description:	The result code for the event
Descriptor Type:	typeLongInteger
Required or Optional?	Optional (The absence of a keyErrorNumber parameter in the reply indicates that the event was handled successfully.)

Notes

The destination table is optional and should be created and returned as the direct object if not specified. The keyAEInsertHere specifies where to put the newly created rows.

Since the functions are specified by a four character code, both the client application and the server must have a list of common functions to specify. There is no provision for textual functions or for code resources since this is intended to be platform independent. Instead, the table of common functions should be extended.

Result Codes

errAEEventFailed	–10000	The Apple event handler failed when attempting to handle the Apple event.
errAENoSuchGroupFunction	–10018	The keyAEGroupFunctions parameter is not a known value.

Sort—order an object by one or more of its elements

The Sort Apple event is used to order an object by one or more of its elements. For example, a client application may wish to sort a table based on one of its columns or a selection of rows based on a cell or column. The sort type indicates the way by which to sort, such as ascending or descending order. The element by which to sort may be a list of items; in this case, the first element of the list is the most significant item on which to sort, and each subsequent item is a less significant item. Applications which do not support multiple item sorts can either perform multiple sorts in response to this event or simply ignore all but the first sort element in the keyAESortElement list.

Event Class kAEDatabase

Event ID kAESort

Parameters

keyDirectObject

Description:	The object to sort. (The value must be one of the following: cTable, cRowSelection, or a list of cRows.)
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Required

keyAESortElement

Description:	The columns by which to sort (may be a list)
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Required

keyAESortType

Description:	Sort type (may be a list)
Descriptor Type:	typeShortInteger
Required or Optional?	Optional

Reply Parameters None

Notes If keyAESortElement is an optional parameter, then the client application can instruct server applications to unsort a sorted table by omitting or sending a null parameter. The keyAESortType parameter would be ignored in this case. If the keyAESortElement is a list, and the keyAESortType is a scalar, then the sort type specified is used for all fields that are to be sorted.

If there is a list of sort types specified as keyAESortType, then there must be a matching list in the parameter keyAESortElement and each sort element must be sorted by the corresponding sort type. The default sort should be an ascending textual sort.

Result Codes

errAEEventFailed	–10000	The Apple event handler failed when attempting to handle the Apple event.
errAENoSuchSortType	–10017	The keyAESortType parameter is not a known value.

Object classes defined in the Database suite

The Apple event object classes defined in the Database suite are described in the following sections. Table 2 lists these object classes.

■ **Table 2** Apple event object classes defined in the Database suite

Object class ID	Description
cCell	A cell from a table *Properties:* pBestType, pClass, pDefaultType, pFormula, pLock, pName, pProtection, pRepeatsize, pValue *Element Classes:* None
cColumn	A column of cells from a table *Properties:* pAccess, pBestType, pClass, pDefaultType, pFormula, pLock, pName, pNullsOk, pProtection, pRepeating, pRepeatSize, pUniqueValue *Element Classes:* cCell, cColumn
cDatabase	A database *Properties:* pAccess, pBestType, pClass, pDefaultType, pLock, pName *Element Classes:* cSession, cTable
cDBMS	A database management system *Properties:* pBestType, pClass, pDefaultType, pName *Element Classes:* cDatabase, cSession
cHost	A network host containing a DBMS or database *Properties:* pBestType, pClass, pDefaultType, pName *Element Classes:* cDatabase, cDBMS, cSession
cKey	An indexed column *Properties:* pBestType, pClass, pCurrentSort, pDefaultType, pName, pPrimaryKey, pUniqueValue *Element Classes:* cColumn

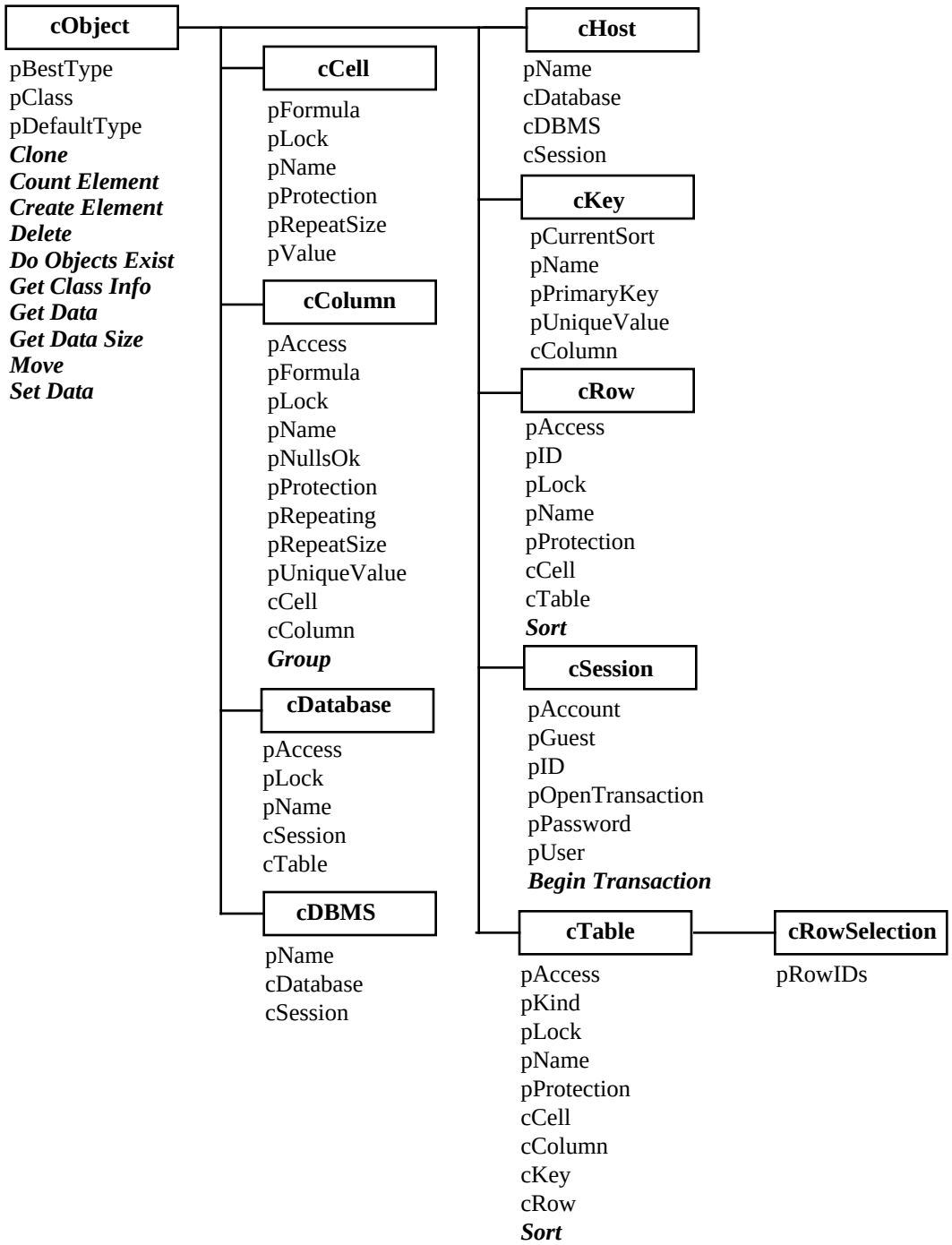
(continued)

■ **Table 2** Apple event object classes defined in the Database suite *(continued)*

Object class ID	Description
cRow	A row of cells from a table *Properties:* pAccess, pBestType, pClass, pDefaultType, pID, pLock, pName, pProtection *Element Classes:* cCell, cTable
cRowSelection	A saved selection of rows *Properties:* pAccess, pBestType, pClass, pDefaultType, pKind, pLock, pName, pProtection, pRowIDs *Element Classes:* cColumn, cKey, cRow
cSession	An active session connected to a host, DBMS, or database *Properties:* pAccount, pBestType, pClass, pDefaultType, pGuest, pID, pOpenTransaction, pPassword, pUser *Element Classes:* None
cTable	A table of rows, columns, and cells from a database *Properties:* pAccess, pBestType, pClass, pDefaultType, pKind, pLock, pName, pProtection *Element Classes:* cCell, cColumn, cKey, cRow

Figure 1 illustrates the inheritance hierarchy for the object classes defined in the Database suite. Listed for each object class are the properties, element classes, and Apple events that have not been inherited from object classes higher in the inheritance hierarchy.

■ **Figure 1** Object inheritance hierarchy for the Database suite



cCell—a cell from a table

The cCell object class is the class for atomic data in a DBMS. It is an extension of the cCell object class defined in the Table suite. The intersection of a row and a column is called a *cell*. The data contained in a cell is called atomic data. The contents of a cell may be of any data type, however all cells in one column typically have the same data type. This data type can be accessed through the pDefaultType property and can be returned as cText by a coercion handler.

Superclass cObject (Core suite)

Default Descriptor Type typeCell

Properties

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pFormula

Description:	The formula for the cell (inherits or overrides the column's formula, if any)
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pLock

Description:	The lock status of the object in the current transaction (The value must be one of the following: kExclusiveLock, kNoLock, or kSharedLock.)
Object Class ID:	enumLockTypes
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pName

Description:	The name of the cell
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pProtection

Description:	Specifies whether the elements or pFormula property of the cell can be changed (The value must be one of the following: kAEFormulaProtect, kAEReadOnly, or kAEReadWrite.)
Object Class ID:	enumProtection
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pRepeatSize

Description:	Indicates the actual number of values for the particular cell
Object Class ID:	cLongInteger
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pValue

Description:	The data in the cell
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes None

Apple Events *Apple events from the Core suite:*

Do Objects Exist	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Set Data	Inherited from cObject

Notes The actual number of values for a particular cell is indicated by the pRepeatSize property of cCell, while the pValue property contains a list of values for that cell. The pRepeatSize property reports the number of items in a list, while the pDefaultType property reports the type of elements in the list.

Locking is used by a database in order to prevent two or more client applications from making simultaneous changes to stored data. Locking requires a client application to obtain an exclusive lock on a piece of data before being allowed to modify it. A client application can also request a shared lock in order to prevent other users from obtaining an exclusive lock. A shared lock does not allow the requesting client application to modify the data. For example, suppose one client application wants to run a report which requires each row to be read exactly once, in the order of a particular column. The client application can request a shared lock on the table object that will prevent any exclusive locks from being granted on the table object or any other objects contained in the table. A lock always applies to the locked object and to any objects contained in the locked object. Linkset tables that are contained in a row are an exception to this. (Refer to the section describing the cRow object class for the definition of linkset.)

Locking a row should not lock every linked row, rather it should lock only the cells in the row. Once the table is locked, the client application can begin reading the rows with the assurance that no other client application will delete, or modify the rows in the table.

The usual procedure is to lock a row before reading or updating it. If required, a client application can also lock individual cells, columns, whole tables, or databases. If a user is browsing through records, the client application can obtain a shared lock on the record that is being browsed and later upgrade the lock status to exclusive when the user decides to edit the record. To lock an object the client application simply sets the pLock property of the object.

Timeout or deadlock errors may be returned in response to a request to lock an object. A *deadlock* occurs when two users try to lock the same two objects in the reverse order. For example, if user A locks object A and user B locks object B, it would not be possible for user A to lock object B because it was already locked by user B. It follows that user B will also encounter the same problem if it attempts to lock object A. The server resolves this problem by detecting the deadlock at the point where the last lock was requested.

A timeout error occurs whenever a lock request issued by a client application is not used or referenced by the client application within a certain time frame. This causes the server to give up on the client application and issue a timeout error. Timing out a lock request has its own drawback. If the timeout is too short, the client application may just retry the lock request. This in turn may prevent the server from seeing the deadlock, because the cycle is always one request short. Because of this, database applications are encouraged to abort a transaction whenever a lock request is canceled. For example, when a Set Data Apple event is performed on a locked property, it could result in a request that returns an error message. In such a case, the application should abort the transaction and start over.

Client applications cannot see the locked status obtained by other client applications. This is to discourage client applications from implementing their own polling mechanism for obtaining locks, such as polling until the status becomes kNoLock. This is not allowed because the server must maintain a complete list of pending lock requests in order to detect any deadlocks.

On the other hand, a client application can read the pLock property of an object to find out the locked status of that object with respect to a particular client application or transaction. Only locks that were set by the client application are revealed; otherwise kNoLock is returned.

Locks are used during transactions that operate on the data stored in a database. Since the data must be locked before it is changed, locks are usually obtained in the context of a transaction. If a lock is obtained in the context of a transaction, the server will normally require that the locks be retained until the transaction either ends normally or aborts. Thus, any attempt to release a lock by setting the pLock property to kNoLock will be ignored without returning an error. Furthermore, when the transaction ends or is aborted, the server will automatically release all locks obtained during the transaction.

If a lock is obtained outside the context of any transaction, the lock must be released by setting pLock to kNoLock.

The possible error conditions are:

- errAEDeadLock - a deadlock was detected. The transaction is aborted.
- errAELockRequestTimeout - a lock request timed out. The transaction is aborted.
- errAELockRequestTimeout - should be returned if a transaction remains inactive for a long period of time without any pending lock request, as in the case of the client application that has failed to continue the transaction, and no possible deadlocks exist. A well designed server would return this error only if the inactive transaction is blocking a lock request from an active transaction. This is a good approach for handling deadlocks that are not detected properly by the server, such as situations where the client applications are polling database attributes on their own.

cColumn—a column of cells from a table

The cColumn object class is the class for the representation of a single column in a table.

Superclass cObject (Core suite)

Default Descriptor Type typeColumn

Properties

pAccess

Description:	Access privileges. (The value must be one of the following or an additive combination of: kCreateAccess, kDeleteAccess, kReadAccess, kUpdateAccess, or kWriteAccess.)
Object Class ID:	enumAccess
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pFormula

Description:	The formula for the cell (inherits or overrides the column's formula, if any)
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pLock

Description:	The lock status of the object in the current transaction. (The value must be one of the following: kExclusiveLock, kNoLock, or kSharedLock.)
Object Class ID:	enumLockTypes
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pName

Description:	The name of the column
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pNullsOk

Description:	Indicates whether nulls are allowed in the column
Object Class ID:	cBoolean
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pProtection

Description:	Specifies whether the elements or pFormula property of the cell can be changed. (The value must be one of the following: kAEFormulaProtect, kAEReadOnly, or kAEReadWrite.)
Object Class ID:	enumProtection
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pRepeating

Description:	Indicates whether it is a repeating column. (The value must be one of the following: kAEFixedRepeat, kAESingleValued, or kAEVariableRepeat.)
Object Class ID:	enumRepeatValues
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pRepeatSize

Description:	Specifies the maximum number of values that can be stored in each cell of a column. For fixed repeat columns, this property indicates the number of values that can be stored in each cell of a column, whereas, for variable repeat columns, it indicates the maximum possible number of values that can be stored in a column.
Object Class ID:	cLongInteger
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pUniqueValue

Description:	Indicates whether the values in this column have to be unique
Object Class ID:	cBoolean
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

Element Classes

cCell

Description:	Cells in the column
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cColumn

Description:	Columns contained in a *grouped column*, that is, a column that contains a number of other columns.
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

Apple Events

Apple events from the Core suite:

Clone	Inherited from cObject
Count Elements	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Move	Inherited from cObject
Set Data	Inherited from cObject

Apple events from the Database suite:

Group	Not inherited

Notes

A column may be specified as a repeating column, a grouped column, or a repeating grouped column. A *repeating column* is a column that can contain a list of values instead of a single value. There are three types of repeating columns: kAESingleValued, which is a single number; kAEFixedRepeat, which is a column with the same number of values in each cell; and kAEVariableRepeat which is a column where each cell can have a different number of values. A *grouped column* contains no data itself but contains a number of other columns. A *repeating grouped column* contains a list of values for a number of other columns.

cDatabase—a database

The cDatabase object class is the class for database tables, each of which contains rows and columns.

Superclass cObject (Core suite)

Default Descriptor Type typeAEDescList

Properties

pAccess

Description:	Access privileges. (The value must either be one of the following or an additive combination of: kCreateAccess, kDeleteAccess, kReadAccess, kUpdateAccess, or kWriteAccess.)
Object Class ID:	enumAccess
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pLock

Description:	The lock status of the object in the current transaction. (The value must be one of the following: kExclusiveLock, kNoLock, or kSharedLock.)
Object Class ID:	enumLockTypes
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pName

Description:	The name of the cell
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes

cSession

Description:	Active sessions using this database
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	None

cTable

Description:	Represents tables in the database
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

Apple Events

Apple events from the Core suite:

Clone	Inherited from cObject
Count Elements	Inherited from cObject
Create Element	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Set Data	Inherited from cObject

Notes

The Get Class Info Apple event should be sent to determine if a database needs a session. If the database can contain a session but none are present, then the client application must create a session with the Create Element Apple event. If the database needs a different session than its containing DBMS, then it should require a new cSession object; otherwise one should not be listed in Get Class Info. If the database cannot contain a session object, then no session is necessary to use the database.

cDBMS—a database management system

The cDBMS object class is the class for a particular brand of database management system.

Superclass cObject (Core suite)

Default Descriptor Type typeAEDescList

Properties

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pName

Description:	The name of the cell
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes

cDatabase

Description:	Databases in the DBMS
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cSession

Description:	Active sessions using this DBMS
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	None

Apple Events

Apple events from the Core suite:

Count Elements	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Set Data	Inherited from cObject

Notes

Some DBMS brands do not require any specific connection and can use the cSession information from their cHost object if desired. Other DBMS brands may require the user to log in. Determine this with a Get Class Info event on the cHost object. If cSession is present, then send a Create Element event with a cSession object containing user name information. Once a DBMS is selected, the databases accessible by that DBMS brand can be selected.

Some servers may not require a DBMS brand to be selected. If a server contains no DBMSs, the container for the database may be specified as the host or NULL if there are no hosts. If a server contains only one DBMS brand, then that DBMS should be specified as the container for any databases.

cHost—a network host containing a DBMS or database

The cHost object class is the class for a computer that runs one or more DBMS brands. If the DBMS is being accessed over a network, the server becomes the network host.

Superclass cObject (Core suite)

Default Descriptor Type typeAEDescList

Properties

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pName

Description:	The name of the host
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes

cDatabase

Description:	Databases in this host
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cDBMS

Description:	DBMS brands handled by this host
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cSession

Description:	Active sessions using this host
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	None

Apple Events *Apple events from the Core suite:*

Count Elements	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Set Data	Inherited from cObject

Notes Some hosts do not require any specific connection. Other hosts may require the user to log in. Determine this with the Get Class Info event on the cHost object. If cSession is present, send a Create Element event with a cSession object containing user name information. Once a host is selected, the DBMS brands or databases contained within that host can be selected.

Some servers may not require a host to be selected. If a server contains no hosts, the container for a DBMS may be specified as NULL. If a server contains only one host, then that host should be specified as the container for any DBMS.

Some servers may not require a DBMS brand to be selected. If a server contains no DBMSs, the container for a database may be specified as the host or NULL if there are no hosts.

cKey—an indexed column

The cKey object class is the class for a collection of indexed columns that can be searched or sorted faster than regular columns.

Superclass cObject (Core suite)

Default Descriptor Type typeKey

Properties

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pCurrentSort

Description:	Sort type constant
Object Class ID:	cShortInteger
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pName

Description:	The name of the key
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pPrimaryKey

Description:	Indicates that this key is the primary key for a table. There are no requirements for a primary key to exist, but for any table only one primary key may exist. If pPrimaryKey is TRUE, then the key should also be unique with pUniqueValue as TRUE.
Object Class ID:	cBoolean
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pUniqueValue

Description:	Indicates if the key is guaranteed to be unique for every row in the table. If pUniqueValue is TRUE, then any attempt to create a new row with duplicate values or to modify an existing row to contain duplicate values will result in an errNotUnique error.
Object Class ID:	cBoolean
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes

cColumn

Description:	Columns in the key
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

Apple Events

Apple events from the Core suite:

Clone	Inherited from cObject
Count Elements	Inherited from cObject
Create Element	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Move	Inherited from cObject
Set Data	Inherited from cObject

Notes

The primary key for a table is a column or a combination of columns that uniquely identifies each row. The primary key is almost always indexed and is the preferred key for relational databases. There may be many different ways by which one can uniquely identify a row, hence there may be many possible candidates for the primary key. However, only one key can be the primary key. If more than one key satisfies the uniqueness requirement for a table, then the server must make an arbitrary decision as to which key will be the primary key.

Databases that do not support key fields should simply return a value equivalent to zero for the number of keys in a table and fail when asked to create keys. Since a key can contain multiple columns, this is the way to specify concatenated keys. The column elements should appear in their order of importance. The pCurrentSort property should contain the same constant used in the Sort event.

cRow—a row of cells from a table

The cRow object class is the class for a single record in a table.

Superclass cObject (Core suite)

Default Descriptor Type typeRow

Properties

pAccess

Description:	Access privileges. (The value must be one of the following or an additive combination of: kCreateAccess, kDeleteAccess, kReadAccess, kUpdateAccess, or kWriteAccess.)
Object Class ID:	enumAccess
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pID

Description:	A unique ID for this row. Internally the ID property can be of any desired type. The Object Class ID is listed as cText so that the ID can be handled by scripting systems. Most of the common ID types should be easily coerced to and from text.
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pLock

Description:	The lock status of the object in the current transaction. (The value must be one of the following: kExclusiveLock, kNoLock, or kSharedLock.)
Object Class ID:	enumLockTypes
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pName

Description:	The name of the row
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pProtection

Description:	Specifies whether the elements or pFormula property of the cell can be changed. (The value must be one of the following: kAEFormulaProtect, kAEReadOnly, or kAEReadWrite.)
Object Class ID:	enumProtection
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes

cCell

Description:	Cells in the column
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cTable

Description:	Name of the linkset
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

Apple Events

Apple events from the Core suite:

Clone	Inherited from cObject
Count Elements	Inherited from cObject
Create Element	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Move	Inherited from cObject
Set Data	Inherited from cObject

Apple events from the Database suite:

Sort	Not inherited

Notes

Each row of a table is structured identically, except that variable repeating columns can have a different repeat value for each row. Rows may contain a table of other rows, representing *child* rows linked to a *parent* row. However, the name of the cTable object contained in a cRow object is the same as the name of the linkset and not the name of the child row's container.

A *linkset* is an abstract concept that can be defined as a single parent row with zero or more related child rows. To understand the concept of linksets, consider the example of a customer-invoice relational database that consists of an invoices table and a customer table. The customer table has a primary "customer account," and the invoice table has a column that refers to the customer by his or her account. To relate an invoice and a customer, the database stores the customer's account number in the customer account column of the invoice row. Whenever data is to be accessed, the database can easily locate the customer row by using the customer account number. Each customer record will "own" a linkset, where the child records in

each linkset are those records that are related to the particular parent record.

In the case of a network database, records are related through explicit operations that add or remove child records from a parent record. These operations do not create or delete any records, instead, they maintain the relationships between the records. The implementation may be a linked list of child records. In the case of the customer-invoice database, each child record invoice has a reference to the next and previous child record in addition to the parent record.

A relational database can map linksets by performing a search operation to find all of the child rows. Note that in a relational database the “linking” field is usually indexed. The customer account field in the invoices table is indexed in order to allow fast retrieval of all invoices related to a particular customer. A network database will simply map the existing sets to the linkset objects.

Linksets are easily mapped into the Database suite by allowing the parent row to contain a table object, which in turn contains each child row. The name of the table is taken from the linkset name, which in turn is taken from the naming convention in a particular database.

Some relational databases will not explicitly identify where the relationship exists. These databases have no linksets and relationships are defined by the users of the database performing explicit find operations, which can also be executed through the Apple event interface.

cRowSelection—a saved selection of rows

The cRowSelection object class is the class that represents an arbitrary selection of rows in a DBMS. Since some selections may be difficult or time consuming to collect, this object is intended as a way to save arbitrary collections of rows.

Superclass cTable (Database suite)

Default Descriptor Type typeRowSelection

Properties

pAccess

Description:	Access privileges. (The value must be one of the following or an additive combination of: kCreateAccess, kDeleteAccess, kReadAccess, kUpdateAccess, or kWriteAccess.)
Object Class ID:	enumAccess
Inherited?	Yes
Modifiable or Non-modifiable?	Non-modifiable

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pKind

Description:	Defines the property as a cursor, linkset, table, or view. (The value must be one of the following: kCursor, kLinkset, kTable, or kView.)
Object Class ID:	enumTableTypes
Inherited?	Yes
Modifiable or Non-modifiable?	Non-modifiable

pLock

Description:	The lock status of the object in the current transaction. (The value must be one of the following: kExclusiveLock, kNoLock, or kSharedLock.)
Object Class ID:	enumLockTypes
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pName

Description:	The name of the table
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pProtection

Description:	Specifies whether the elements or pFormula property of the cell can be changed. (The value must be one of the following: kAEFormulaProtect, kAEReadOnly, or kAEReadWrite.)
Object Class ID:	enumProtection
Inherited?	Yes
Modifiable or Non-modifiable?	Modifiable

pRowIDs

Description:	A list of row IDs for the rows in this selection
Object Class ID:	cAEList
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

Element Classes

cColumn

Description:	Columns in the table
Inherited?	Yes
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cKey

Description:	Key or indexed columns in this table
Inherited?	Yes
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cRow

Description:	Rows in the table
Inherited?	Yes
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

Apple Events

Apple events from the Core suite:

Clone	Inherited from cObject
Count Elements	Inherited from cObject
Create Element	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Set Data	Inherited from cObject

Apple events from the Database suite:

Sort	Not inherited

Notes

To create a cRowSelection object, use a Create Element event with keyAEData parameter being typeRowSelection. The typeRowSelection parameter contains an object specifier that resolves to a set of rows. Once created, a cRowSelection object can be used as a reference to the rows without the tests that were required in the original object specification.

Selections consist of rows in the form of a table. The selection object can be a list of row IDs that are used to reference the actual rows in their original locations. The selection object can also be used as if it were a table. Although the actual implementation is up to the server, a selection should be similar to a table made up of the rows named in the pRowIDs property when the elements are referenced. A selection should also be able to return a list of row IDs when asked for the pRowIDs property.

cSession—an active session connected to a host, DBMS, or database

The cSession object class is the class that describes the user of the host, DBMS, or database. It contains access information, such as the user name, account, and the password used to access the object. It uses a value of type boolean (TRUE/FALSE) to determine whether the session is conducted as an unauthenticated guest or through the user information. The session ID is a unique session identifier.

Superclass cObject (Core suite)

Default Descriptor Type typeSession

Properties

pAccount

Description:	An optional account number
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pGuest

Description:	Whether the user name and account is used to log in or whether the user is signed in at the default guest level
Object Class ID:	cBoolean
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pID

Description:	A unique ID for this session. Internally the ID property can be of any desired type. The Object Class ID is listed as cText so that the ID can be handled by scripting systems. Most of the common ID types should be easily coerced to and from text.
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pOpenTransaction

Description:	Current transaction(s) list
Object Class ID:	cLongInteger
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pPassword

Description:	The password supplied to authenticate this session
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pUser

Description:	The user name used to access the DB
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

Element Classes None

Apple Events

Apple events from the Core suite:

Count Elements	Inherited from cObject
Create Element	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Set Data	Inherited from cObject

Apple events from the Database suite:

Begin Transaction	Not inherited

Notes

One way to reference a session is to use the Begin Transaction event with a session specifier. A session's pOpenTransaction property is undefined until the session object is specified in the Begin Transaction event. The transaction ID is returned by the Begin Transaction event. Since sessions are used for authentication, it does not make sense for a server to allow read access to its data. This means that any request for session properties should return an error. The server may allow writing to the properties if session information can be changed; otherwise, they should only be specified when the session is created.

Every event in a transaction can be authenticated easily without the overhead of a specific session reference. However, there are certain situations that require a session for a specific event that is not part of a transaction. These situations require a session parameter to be added to an arbitrary event. This parameter should be:

keyAESession

Description:	A session reference for authentication
Descriptor Type:	typeObjectSpecifier
Required or Optional?	Optional

If you want these events to be parsed by a scripting system, you must add this parameter to the application for each event on which you use it. This may require you to replicate the entire Core suite, or whatever parts of it you add this parameter to, in your private application; this is why the transaction method is preferred.

Sessions are opened upon creation of the cSession object with the Create Element event. The initial values are provided in the Create Element event's typeSession record, but they may be rejected based on the server's verification process. If the values are rejected, then the session is not created, and the Create Element event returns an error through the keyErrorString parameter. If the Create Element event succeeds, then the returned object specifier can be saved and used for future references to this session.

Some objects may not require a session element. Use the Get Class Info event or an object that has a cSession element to see if it actually requires one. If it does, you must create the session object with Create Element before using the object. All of the properties of a session are optional, depending on the server's needs. A Get Class Info event should always be sent to determine what information is necessary to validate a session. Sessions may vary from object to object; a cHost session may be different than a cDatabase session.

If a session parameter is expected on any event but not present, the server should attempt to treat the request as a Guest access with the corresponding public privileges if available. Guest sessions can also be requested explicitly by creating a session with the pGuest property initialized to TRUE. The server must return errAEPrivilegeError in the reply keyword keyErrorNumber and a message in the keyErrorString keyword if Guest sessions are not allowed or if any specified session data is invalid.

The session ID property is documented as text since any ID should be representable as text. A server can use any data type for the session ID since the server is typically the only one referencing it. All client applications should use the object reference to the session that is returned by Create Element to refer to the session. Although this is a simple object specifier referring to the session by ID, it is opaque to a client application. A client application only sees an arbitrary object specifier.

cTable—a table of rows, columns, and cells from a database

The cTable object class is the class for tables. A table is a collection of rows and columns in a database.

Superclass cObject (Core suite)

Default Descriptor Type typeTable

Properties

pAccess

Description:	Access privileges. (The value must be one of the following or an additive combination of: kCreateAccess, kDeleteAccess, kReadAccess, kUpdateAccess, or kWriteAccess.)
Object Class ID:	enumAccess
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pBestType

Description:	The descriptor type that can contain the most information from objects of this object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pClass

Description:	The four-character class ID for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pDefaultType

Description:	The default descriptor type for the object class
Object Class ID:	cType
Inherited?	Yes, from cObject
Modifiable or Non-modifiable?	Non-modifiable

pKind

Description:	Defines the property as a cursor, linkset, table, or view. (The value must be one of the following: kCursor, kLinkset, kTable, or kView.)
Object Class ID:	enumTableTypes
Inherited?	No
Modifiable or Non-modifiable?	Non-modifiable

pLock

Description:	The lock status of the object in the current transaction. (The value must be one of the following: kExclusiveLock, kNoLock, or kSharedLock.)
Object Class ID:	enumLockTypes
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pName

Description:	The name of the table
Object Class ID:	cText
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

pProtection

Description:	Specifies whether the elements or pFormula property of the cell can be changed. (The value must be one of the following: kAEFormulaProtect, kAEReadOnly, or kAEReadWrite.)
Object Class ID:	enumProtection
Inherited?	No
Modifiable or Non-modifiable?	Modifiable

Element Classes

cCell

Description:	Cells in the table
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cColumn

Description:	Columns in the table
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cKey

Description:	Key or indexed columns in this table
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

cRow

Description:	Rows in the table
Inherited?	No
Modifiable or Non-modifiable?	Modifiable
Key Forms:	formAbsolutePosition, formName, formPropertyID, formRange, formRelativePosition, formTest

Apple Events

Apple events from the Core suite:

Clone	Inherited from cObject
Count Elements	Inherited from cObject
Create Element	Inherited from cObject
Delete	Inherited from cObject
Do Objects Exist	Inherited from cObject
Get Class Info	Inherited from cObject
Get Data	Inherited from cObject
Get Data Size	Inherited from cObject
Move	Inherited from cObject
Set Data	Inherited from cObject

Apple events from the Database suite:

Sort	Not inherited

Descriptor types defined in the Database suite

The descriptor types defined in the Database suite are described in the following sections. Table 3 lists these descriptor types.

■ **Table 3** Descriptor types defined in the Database suite

Descriptor type	Description
typeCell	A cell object
typeColumn	A column object
typeKey	A key fields object
typeRow	A row object
typeRowSelection	A row selection object
typeSession	A session object
typeTable	A table object

typeCell—a cell object

A typeCell descriptor record contains data to create or retrieve a cCell object.

Description To create a typeCell descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeCell descriptor record.

Keyword	Descriptor type	Description
keyAEData	typeAEList	A list of values in the cell with the format described by keyAEDefaultType
keyAEDefaultType	typeType	The default data type
keyAEFormula	typeChar	The cell's formula
keyAEName	typeChar	The name of the cell
keyAEProtection	typeEnumeration	The formula's protection
keyAERepeatSize	typeLongInteger	The number of values in keyAEData list

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

typeColumn—a column object

A typeColumn descriptor record contains the data needed to create or retrieve a cColumn object.

Description To create a typeColumn descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeColumn descriptor record.

Keyword	Descriptor type	Description
keyAEAccess	typeEnumeration	Access privileges
keyAECellList	typeAEList	The column values and formulas as a list of typeCell descriptors
keyAEFormula	typeChar	The formula for the column
keyAEName	typeChar	The name of this row
keyAENullsOK	typeBoolean	Indicates whether Nulls are allowed
keyAEProtection	typeEnumeration	The formula privileges
keyAERepeating	typeEnumeration	Indicates whether it is a repeating column. (kAEFixedRepeat, kAESingleValued, or kAEVariableRepeat.)
keyAERepeatSize	typeLongInteger	Number of times to repeat
keyAEUniqueValue	typeBoolean	Indicates whether values in this column have to be unique

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

typeKey—a key fields object

A typeKey descriptor record contains the data to create or retrieve a cKey object.

Description To create a typeKey descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeKey descriptor record.

Keyword	Descriptor type	Description
keyAEColumnList	typeAEList	A list of column object specifiers
keyAECurrentSort	typeShortInteger	The current sort constants
keyAEName	typeChar	The name of the key
keyAEPrimaryKey	typeBoolean	Indicates whether the key is the primary key
keyAEUniqueValue	typeBoolean	Indicates whether the key is unique

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

typeRow—a row object

A typeRow descriptor record contains data to create or retrieve a cRow object class.

Description To create a typeRow descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeRow descriptor record.

Keyword	Descriptor type	Description
keyAEAccess	typeEnumeration	The access privileges
keyAECellList	typeAEList	The row values and formulas as a list of typeCell descriptors
keyAEID	typeChar	The permanent ID number for this row
keyAEName	typeChar	The name of this row
keyAEProtection	typeEnumeration	The formula privileges

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

typeRowSelection—a row selection object

A typeRowSelection descriptor record contains the data to create or retrieve a cSession object. Information about existing sessions cannot be obtained from a typeRowSelection descriptor record.

Description To create a typeRowSelection descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeRowSelection descriptor record.

Keyword	Descriptor type	Description
keyAEAccess	typeEnumeration	The access privileges
keyAEData	typeObjectSpecifier	An object specifier that resolves to a set of rows to be referenced in this row selection
keyAEKind	typeEnumeration	The kind of table
keyAEName	typeChar	The name of this row selection
keyAEProtection	typeEnumeration	The table's protection value

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

typeSession—a session object

A typeSession descriptor record contains the data necessary to create a cSession object. Note that information about existing sessions cannot be obtained.

Description To create a typeSession descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeSession descriptor record.

Keyword	Descriptor type	Description
keyAEAccount	typeChar	An optional account name or number for login verification
keyAEGuest	typeBoolean	Whether or not to validate the session at the default guest level
keyAEPassword	typeChar	The session password
keyAEUser	typeChar	The user name to access this object's container

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

Notes The keyAEGuest field is used to determine if the user, account, and password fields are required. If keyAEGuest is TRUE, then they are not used and do not need to be included or referenced. The session should be validated at whatever the default guest access level is. If keyAEGuest is false, then the fields keyAEUser and keyAEPassword (and optionally keyAEAccount) should be used for session verification.

The keyAEAccount field is an optional field typically used for logging onto mainframes. It can be omitted if it is not needed.

The keyAEPassword field is a password that is used to verify the session. If the password does not match the password on the server for that particular user, then the session creation should fail.

typeTable—a table object

A typeTable descriptor record contains data to create or retrieve a cTable object.

Description To create a typeTable descriptor record, you coerce an Apple event record containing the following fields into the equivalent typeTable descriptor record.

Keyword	Descriptor type	Description
keyAEAccess	typeEnumeration	The access privileges
keyAEColumns	typeAEList	A list of typeColumn descriptors
keyAEData	typeAEList	The data in the table as a list of rows, with each row as a list of typeCell descriptors
keyAEKind	typeEnumeration	The kind of table
keyAEName	typeChar	The name of this table
keyAEProtection	typeEnumeration	The table's protection value
keyAERowList	typeAEList	A list of typeRow descriptors

Note that the Apple Event Manager can coerce any Apple event record into any other descriptor type. A special coercion handler is not required.

Data Size Variable

Notes The keyAERowList and keyAEColumns descriptors should not include the keyAEData keyword. The keyAEData keyword should only appear in the typeTable descriptor, and the data should be a list of lists. Each element in the list is a row, and that row is a list of typeCell descriptors containing only the unique information for that cell.

Key forms defined in the Database suite

Table 4 lists the key forms defined in the Database suite. The italicized words in each example correspond to the key (the portion of the object specifier record that distinguishes an object from other objects of the same class in the same container). For more information about keys and key forms, see the Apple Event Manager chapter of *Inside Macintosh*: *Interapplication Communication*.

■ **Table 4** Key forms defined in the Database suite

Key form constant	Description
formAbsolutePosition	Specifies the position of an element in relation to the beginning or end of its container (for example, "word *5* of . . . "), or specifies one or more elements with a constant defined in the Apple Event Manager chapter of *Inside Macintosh*: *Interapplication Communication*, such as kAEFirst (for example, "the *first* word in paragraph 12 . . . ") or kAEAll (for example, "*all* the words in paragraph 12 . . . ")
formName	Specifies an element by its name (for example, "the document *named 'MyDoc' "*)
formPropertyID	Specifies a property of an object by its four-character property ID (for example, "*the font* of word 1")
formRange	Specifies a list of elements between two other elements (for example, "the words *between 'Wild' and 'Zanzibar,' inclusive*")
formRelativePosition	Specifies an element immediately before or after a container (for example, "*the next* word *after* the word whose style is bold")
formTest	Specifies one or more elements that pass a test; values of one or more properties or elements are tested (for example, "the first paragraph *that is centered and that begins with the word 'Wild' "*)

Comparison operators defined in the Database suite

Table 5 lists the comparison operators defined in the Database suite.

■ **Table 5** Comparison operators defined in the Database suite

Comparison operator constant	Operator	Description
kAEBeginsWith	`'bgwt'`	The value of the first operand begins with the value of the second operand (for example, the string “operand” begins with the string “opera”)
kAEContains	`'cont'`	The value of the first operand contains the value of the second operand (for example, the string “operand” contains the string “era”)
kAEEndsWith	`'ends'`	The value of the first operand ends with the value of the second operand (for example, the string “operand” ends with the string “and”)
kAEEquals	`'=  '`	The value of the first operand is equal to the value of the second operand
kAEGreaterThan	`'>  '`	The value of the first operand is greater than the value of the second operand
kAEGreaterThanEquals	`'>= '`	The value of the first operand is greater than or equal to the value of the second operand
kAELessThan	`'<  '`	The value of the first operand is less than the value of the second operand
kAELessThanEquals	`'<= '`	The value of the first operand is less than or equal to the value of the second operand

Constants defined in the Database suite

Table 6 lists the constants defined in the Database suite.

■ **Table 6** Constants defined in the Database suite

Constant	Value	Constant	Value
cDatabase	'cDB '	kAESort	'SORT'
cDBMS	'cDBM'	kAEVariableRepeat	'rVar'
cHost	'cHST'	kAscending	$0000
cKey	'cKEY'	kAverage	'AVRG'
cRowSelection	'crsl'	kCount	'CONT'
cSelection	'csel'	kCreateAccess	$0008
cSession	'cSES'	kCursor	'CURS'
enumAccess	'accs'	kDeleteAccess	$0010
enumGroupFunctions	'grup'	kDescending	$0001
enumLockTypes	'lock'	kExclusiveLock	'EXLK'
enumProtection	'prtn'	keyAEAccess	'pACS'
enumRepeatValues	'erpt'	keyAEAccount	'pACT'
enumSortDirection	'sort'	keyAEColumnList	'kCol'
enumTableTypes	'tblt'	keyAEColumns	'COLS'
errAEDeadLock	–10019	keyAECurrentSort	'pSRT'
errAELockRequestTimeout	–10020	keyAEDefaultType	'deft'
errAENoSuchGroupFunction	–10018	keyAEGroupColumns	'GRPC'
errAENoSuchSortType	–10017	keyAEGroupFunctions	'GRPF'
errAENotUnique	–10022	keyAEGuest	'pGST'
errAETransactionTimeout	–10021	keyAEID	'ID '
kAEAbortTransaction	'ABRT'	keyAEKind	'pKND'
kAEDatabase	'DATA'	keyAEName	'pnam'
kAEDBSuite	'dbst'	keyAENullsOK	'pNLS'
kAEFixedRepeat	'rFxd'	keyAEPassword	'pPAS'
kAEFormulaProtect	'fpro'	keyAEPrimaryKey	'pPKy'
kAEGroup	'GRUP'	keyAERepeating	'pRPT'
kAEModifiable	'modf'	keyAERepeatSize	'pRPS'
kAENonModifiable	'nmod'	keyAERowList	'krls'
kAESingleValued	'rSgl'	keyAESession	'SESN'

(continued)

■ **Table 6** Constants defined in the Database suite *(continued)*

Constant	Value	Constant	Value
keyAESortElement	'SRTE'	pCurrentSort	'pSRT'
keyAESortType	'SRTT'	pFormula	'pfor'
keyAEUniqueValue	'pUNQ'	pGuest	'pGST'
keyAEUser	'pUSR'	pID	'ID '
kLinkset	'LINK'	pKind	'pKND'
kMaximum	'MAX '	pLock	'pLCK'
kMean	'MEAN'	pNullsOk	'pNLS'
kMinimum	'MIN '	pOpenTransaction	'pTRN'
kNoAccess	$0000	pPassword	'pPAS'
kNoLock	'NOLK'	pPrimaryKey	'pPKy'
kNumeric	$0002	pRepeating	'pRPT'
kReadAccess	$0001	pRepeatSize	'pRPS'
kSharedLock	'SHLK'	pRowIDs	'pRWS'
kStdDev	'STDV'	pUniqueValue	'pUNQ'
kSum	'TOTL'	pUser	'pUSR'
kTable	'TABL'	pValue	'vlue'
kTextual	$0000	typeCell	'ccel'
kUpdateAccess	$0004	typeColumn	'ccol'
kView	'VIEW'	typeKey	'cKEY'
kWriteAccess	$0002	typeRow	'crow'
pAccess	'pACS'	typeRowSelection	'crsl'
pAccount	'pACT'	typeSession	'cSES'

Index

THE APPLE PUBLISHING SYSTEM

This Apple manual was written, edited, and composed on a desktop publishing system using Apple Macintosh computers and Microsoft Word software. Proof pages were created on an Apple LaserWriter IINTX printer. Final pages were created on the Varityper VT600 imagesetter. PostScript®, the page-description language for the LaserWriter, was developed by Adobe Systems Incorporated.

Text type and display type are Palatino®. Bullets are ITC Zapf Dingbats®. Some elements, such as program listings, are set in Apple Courier.

Writer: Amr Eissa
Illustrator: Janet Anders