New Technical Notes

Macintosh

Developer Support

Data Access Extensions

Networking **M.NW.DAMExtensions**

Written by: Chuq Von Rospach and Dan Strnad May 1992

This Technical Note discusses coding data access extensions that provide an interface between the Data Access Manager and remote data sources. Each of the functions that a data access extension must implement is described.

Introduction

A data access extension is a program that provides an interface between the Data Access Manager and the remote data source. The data access extension implements all of the low-level functions and handles network communication for the Data Access Manager. Because the data access extension implements the low-level Data Access Manager functions, it must return appropriate result codes and handle asynchronous execution of functions as appropriate.

Note: Each data access extension contains a flag that indicates to the Data Access Manager whether the data access extension supports asynchronous execution of routines. If an application attempts to make an asynchronous call to a data access extension that has the first bit (bit 0) of the flags field cleared to 0, the Data Access Manager returns a result code of `rcDBAsyncNotSupp` and terminates execution of the routine. To ensure compatibility of your data access extension with all applications, your data access extension should support asynchronous execution of functions. The data access extension flags field is described in the next section, “Contents of a Data Access Extension.”

As soon as the data access extension begins execution of an asynchronous routine, it should return a `noErr` result code for the function result and set the result field of the asynchronous parameter block to 1. It should return control to the calling routine as quickly as possible. When it terminates execution of the routine, the data access extension must place the return code in the result field. Result codes for each of the data access extension routines are listed in the next section. The asynchronous parameter block is described in “Asynchronous Execution of Routines” in the Data Access Manager chapter of *Inside Macintosh* Volume VI, page 8-50.

When the data access extension has completed execution of an asynchronous routine, it must call the application’s completion routine pointed to by the completionProc field of the asynchronous parameter block. The completion routine is described in “Asynchronous Execution of Routines” in the Data Access Manager chapter of *Inside Macintosh* Volume VI, page 8-50.

The data access extension can use the `ddevRef` field in the asynchronous parameter block for its own purposes.

This Tech Note describes each of the functions that a data access extension must implement.

Contents of a Data Access Extension

A data access extension consists of a file of type 'ddev', located in the Extensions Folder. The data access extension file must contain these resources:

'ddev' (function; resource ID is 128)

'STR ' (name of data access extension; resource ID is 128)

'dflg' (version number and flags; resource ID is 128)

The 'ddev' resource contains a function that implements all of the low-level Data Access Manager functions. The Data Access Manager calls the ddev function whenever the manager needs to execute a low-level function.

Here is a function declaration for a ddev function.

```
FUNCTION MyDDev(params: DDEVParams) : OSErr;
```

The params parameter is a parameter block that includes a routine selector. The data

access extension parameter block is described in the next section, “Data Access Extension Parameters.”

You must set bit 6 of the resource attribute byte to 1 for the 'ddev' resource so that the resource is read into the system heap. Resource attributes are discussed in the Resource Manager chapter of of *Inside Macintosh* Volume I.

The 'STR ' resource must contain a character string of not more than 63 characters that specifies the name of the data access extension. Under tsystem 7.0 or later, the data access extension is assumed to reside in the "Extensions" folder within the System folder. The data access extension name is a parameter to the `DBInit` function.

The 'dflg' resource contains two 4-byte fields, as follows:

```
TYPE DDEVFlags =
        RECORD
            version:            LongInt;        {data access extension format}
            flags:              LongInt         {data access extension flags}
        END;
```

The version field indicates the version of the data access extension format used for this data access extension. It must be set to 0 for the version 7.0 Data Access Manager.

The flags field specifies flags that the data access extension must set. At present, only the least significant bit is defined; all other bits must be cleared to 0. Set the flags field to the constant `kAsyncSupported` (that is, set the least significant bit to 1) if this data access extension supports asynchronous calls, or to 0 if it does not. If an application attempts to make an asynchronous call to a data access extension that has the flags field cleared to 0, the Data Access Manager returns a result code of `rcDBAsyncNotSupp`.

Data Access Extension Parameters

This section describes the parameter block that the Data Access Manager passes to a data access extension. The section “Data Access Extension Messages” specifies which parameters are significant for each type of routine and whether each value is passed to the data access extension or returned by the data access extension.

The Data Access Manager passes a parameter block to a data access extension. The parameter block is defined as a `DDEVParams` record.

```
TYPE DDEVParams =
        RECORD
```

```
    message:            Integer;            {routine selector}
        ddevStorage:    LongInt;            {storage for use by }
                                            { data access extension}
        asyncPB:        DBAsyncParmBlkPtr;
                                            {pointer to asynch }
                                            { parameter block}
        sessID:         LongInt;            {session ID}
        returnedID:     LongInt;            {session ID returned}
        version:        LongInt;            {version number}
        start:          LongInt;            {session start time}
        host:           StringPtr;          {name of remote system}
        user:           StringPtr;          {user name}
        password:       StringPtr;          {user password}
        connStr:        StringPtr;          {connection string}
        network:        StringPtr;          {name of the network}
        buffer:         Ptr;                {data buffer}
        err1:           LongInt;            {primary error code returned}
        err2:           LongInt;            {secondary error code }
                                            { returned}
        item1:          StringPtr;          {pointer to object of error }
                                            { message}
        item2:          StringPtr;          {pointer to object of }
                                            { error message}
        errorMsg:       StringPtr;          {pointer to error message}
        timeout:        LongInt;            {timeout value for DBGetItem}
        dataType:       DBType;             {data type}
        sessNum:        Integer;            {session number}
        state:          Integer;            {status of the data source}
        len:            Integer;            {length of data item}
        places:         Integer;            {decimal places in data item}
        flags:          Integer;            {flags}
        abort:          Boolean             {flag for DBBreak}
    END;
```

Field Descriptions

`message` The routine selector that tells the data access extension which function to execute. For the values for this field and descriptions of the routines, see the next section, “Data Access Extension Messages.”

`ddevStorage` Reserved for use by the data access extension. The Data Access Manager sets this field to 0 when it calls the data access extension with the DBOpen message. The data access extension can store any value in this field at that time, and the Data Access Manager retains that value on all subsequent calls to the data access extension. The value of this field does not depend on the session ID; it is the same for all sessions that are using the same data access extension.

`asyncPB` Pointer to the asynchronous parameter block. If the application is making a synchronous call, this field is NIL. The asynchronous parameter block is described in “Asynchronous Execution of Routines” in the Data Access Manager chapter of *Inside Macintosh* Volume VI.

`sessID` The session ID. The data access extension returns the session ID to the DBInit function; all other Data Access Manager functions pass the session ID to the data access extension.

The purpose of the session ID is to provide applications with a unique identifier for each active session. The Data Access Manager reads the session ID returned by the data access extension, and then assigns a unique session ID to each session. The Data Access Manager performs the mapping between the session IDs that it provides to applications and the ones used by each data access extension.

`returnedID` The session ID returned by the DBGetConnInfo function.

`version` The version number of the data access extension assigned by the developer of the data access extension. It is not the same as the version number in the 'dflg' resource of the data access extension, which indicates the format of the data access extension.

`start` The time at which the session was opened, in ticks.

`host` The name of the remote system on which the data source is located.

`user` The name of the user who is establishing a session.

`password` The password associated with the user name.

`connStr` A connection string needed to establish a session.

`network` A string specifying the network in use for this session.

`buffer` A pointer to a buffer containing the item to be sent by the DBSend or DBSendItem functions or received by the DBGetItem function.

`err1` The primary error code returned by the data source.

`err2` The secondary error code returned by the data source.

`item1` A pointer to a NULL-terminated string that identifies the first object of the error message returned by the data source. The use of this parameter depends on the specific data source you are using.

`item2` A pointer to a NULL-terminated string that identifies the second object of the error message returned by the data source. The use of this parameter depends on the specific data source you are using.

`errorMsg` A pointer to the error message returned by the data source.

`timeout` The timeout period for the DBGetItem function, in sixtieths of seconds. When the data access extension executes the DBGetItem function, it requests a data item from the

remote data source. If the remote data source does not return the requested data item in the amount of time specified by the timeout parameter, the data access extension should cancel execution of the DBGetItem function. The timeout value cannot be used if the DBGetItem function is called asynchronously.

`dataType` The data type of a requested or returned data item. Data types are described in “Getting Query Results” in the Data Access Manager chapter of *Inside Macintosh* Volume VI.

`sessNum` The session number. This number is assigned by the data access extension and is unique for all current sessions for a single data access extension only. The same session number can be assigned to concurrent sessions that use different data access extensions.

`state` The status of the data source.

Value	Status
noErr	Execution of a query successful; ready for another
rcDBValue	Output data is available
rcDBError	Execution of a query ended in an error
rcDBExec	Currently executing a query

`len` The length of the data item requested or returned.

`places` The number of decimal places in the data item.

`flags` Flags returned by the DBGetItem function or sent to the DBSendItem function. For the DBGetItem function, if the flags field is set to kDBLastColFlag (that is, the least significant bit is set to 1), the data item is in the last column of the row.

There are no flags currently defined for the DBSendItem function.

`abort` A parameter used by the DBBreak function. The meaning of this parameter depends on the specific implementation of the data source communications system you are using.

Data Access Extension Messages

There are sixteen values that the Data Access Manager can pass to the data access extension in the messages field of the data access extension parameter block. Thirteen of them correspond exactly to the thirteen low-level functions. The other three are used by the Data Access Manager to initialize and close the data access extension and to allow the data access extension to perform routine periodic tasks. The messages that correspond to low-level routines are not described in this section. Instead, only the parameters they use and the result codes they must be able to return are listed. For descriptions of these routines, see the section “Data Access Manager Routines” in the Data Access Manager chapter of *Inside Macintosh* Volume VI. The `DBOpen`, `DBClose`, and `DBIdle` messages are described in detail in this section.

Each parameter in the list is preceded by an arrow that indicates how the parameter is used, as follows:

- → The Data Access Manager passes the value of the parameter as input to the data access extension.
- ← The data access extension returns the value of the parameter after the routine has completed execution.
- ↔ The Data Access Manager provides a value for the parameter, and the data access extension returns another value.

DBOpen

Parameter block

→	00	`message`	word	routine selector; `kDBOpen`
←	02	`ddevStorage`	long	storage for data access extension

When an application calls the `DBInit` function or the `DBStartQuery` function (which calls the `DBInit` function), it specifies a data access extension. If that data access extension is not already in memory, the `DBInit` function loads it into memory and sends it the `kDBOpen` message. The data access extension should allocate any memory it needs at this time. Because the data access extension can be called by more than one application, it should allocate memory in the system heap rather than the application heap. The data access extension can also return a value in the ddevStorage field of the data access extension parameter block.

When the Data Access Manager calls the data access extension, the current resource file is the data access extension file and the default directory is the Extensions Folder on the current startup disk. The data access extension must ensure that both of these values are unchanged when it returns control to the Data Access Manager.

Result codes

`noErr`	0	Data Access Extension initialized successfully
`rcDBError-`	–802	Error initializing data access extension

DBClose

Parameter block

→	00	`message`	word	routine selector; kDBClose
→	02	`ddevStorage`	long	storage for data access extension

When an application calls the `DBEnd` function, closing the last open session for a data access extension, the Data Access Manager follows the `kDBEnd` message with a `kDBClose` message before removing the data access extension from memory. The data access extension should free any memory that it allocated before returning control to the Data Access Manager.

Result code

noErr	0	No error

DBIdle

Parameter block

→	00	message	word	routine selector; kDBIdle
↔	02	ddevStorage	long	storage for data access extension

The Data Access Manager periodically sends the kDBIdle message to each data access extension. The data access extension can ignore this message or take the opportunity to perform periodic tasks. Because the timing of the kDBIdle messages might not be regular, the data access extension must not depend on receiving these messages at particular times or with a particular frequency.

Result code

noErr	0	No error

DBInit

Parameter block

→	00	message	word	routine selector; kDBInit
↔	02	ddevStorage	long	storage for data access extension
→	06	asyncPB	long	pointer to asynch parameter block
←	10	sessID	long	session ID
→	26	host	long	pointer to name of remote system
→	30	user	long	pointer to user name
→	34	password	long	pointer to user password
→	38	connStr	long	pointer to connection string

The DBInit function initiates a session with a remote data source. See “Controlling the Session” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

noErr	0 No error
rcDBError	–802 Error initiating session

DBEnd

Parameter block

→	00	message	word	routine selector; kDBEnd
↔	02	ddevStorage	long	storage for data access extension
→	06	asyncPB	long	pointer to asynch parameter block
→	10	sessID	long	session ID

The DBEnd function terminates a session with a remote data source and terminates the network connection between the application and the remote computer. See “Controlling the Session” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

noErr	0 No error
rcDBError	–802 Error ending session

DBGetConnInfo

Parameter block

→	00	message	word	routine selector; kDBGetConnInfo
↔	02	ddevStorage	long	pointer to storage for ddev
→	06	asyncPB	long	pointer to asynch parameter block
→	10	sessID	long	session ID
←	14	returned ID	long	session ID returned
←	18	version	long	version number
←	22	start	long	session start time in ticks
←	26	host	long	pointer to name of remote system
←	30	user	long	pointer to user name

←	38	connStr	long	pointer to connection string
←	42	network	long	pointer to name of network
→	78	sessNum	word	session number
←	80	state	word	status of data source

The DBGetConnInfo function returns information about the specified session. See “Controlling the Session” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

noErr	0	No error
rcDBBadSessNum	–808	Invalid session number

DBGetSessionNum

Parameter block

→	00	message	word	routine selector; kDBGetSessionNum
↔	02	ddevStorage	long	storage for data access extension
→	06	asyncPB	long	pointer to asynch parameter block
→	10	sessID	long	session ID
←	78	sessNum	word	session number

The DBGetSessionNum function returns a session number. See “Controlling the Session” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

noErr	0	No error
rcDBError	– 802	Error getting session number

DBKill

Parameters used

→	00	message	word	routine selector; kDBKill
↔	02	ddevStorage	long	storage for data access

extension

→	06	asyncPB	long	pointer to asynch parameter block

The DBKill function cancels the execution of an asynchronous call. See “Controlling the Session” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

noErr	0	Asynchronous routine canceled successfully
rcDBError	–802	Error canceling routine

DBSend

Parameter block

→	00	message	word	routine selector; kDBSend
↔	02	ddevStorage	long	storage for data access extension
→	06	asyncPB	long	pointer to asynch parameter block
→	10	sessID	long	session ID
→	46	buffer	long	pointer to data buffer
→	82	len	word	length of data

The DBSend function sends a query or a portion of a query to the remote data source. See “Sending and Executing Queries” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

noErr	0	No error
rcDBError	–802	Error trying to send text

DBSendItem

Parameter block

→	00	message	word	routine selector; kDBSendItem
↔	02	ddevStorage	long	storage for data access

extension

→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID
→	46	`buffer`	long	pointer to data buffer
→	74	`dataType`	long	data type
→	82	`len`	word	length of data item
→	84	`places`	word	decimal places in data item
→	86	`flags`	word	flags

The `DBSendItem` function sends a single data item to the remote data source. See “Sending and Executing Queries” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	No error
`rcDBError`	–802	Error trying to send item

DBExec

Parameter block

→	00	`message`	word	routine selector; `kDBExec`
↔	02	`ddevStorage`	long	storage for data access extension
→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID

The `DBExec` function initiates execution of a query. See “Sending and Executing Queries” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	Execution has begun
`rcDBError`	–802	Error trying to begin execution

DBState

Parameter block

→	00	`message`	word	routine selector; `kDBState`
↔	02	`ddevStorage`	long	storage for data access extension
→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID

The result code returned by the `DBState` function indicates the status of the remote data source. See “Sending and Executing Queries” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	No error; ready for more text
`rcDBValue`	–801	Output data available
`rcDBError`	– 802	Execution ended in an error
`rcDBExec`	–806	Currently executing query

DBGetErr

Parameter block

→	00	`message`	word	routine selector; `kDBGetErr`
↔	02	`ddevStorage`	long	storage for data access extension
→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID
←	50	`err1`	long	primary error code
←	54	`err2`	long	secondary error code
←	58	`item1`	long	pointer to first object of error message
←	62	`item2`	long	pointer to second object of error message
←	66	`errorMsg`	long	pointer to error message

The `DBGetErr` function retrieves error codes and error messages from a remote data source. See “Sending and Executing Queries” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	No error
`rcDBError`	–802	Error retrieving error information

`DBBreak`

Parameter block

→	00	`message`	word	routine selector; `kDBBreak`
↔	02	`ddevStorage`	long	storage for data access extension
→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID
→	88	`abort`	byte	abort flag

The `DBBreak` function can halt execution of a query and reinitialize the remote data source, or it can unconditionally terminate a session with a data source. See “Sending and Executing Queries” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	Execution has begun
`rcDBError-`	–802	Break or abort attempt was unsuccessful

`DBGetItem`

Parameter block

→	00	`message`	word	routine selector; `kDBGetItem`
↔	02	`ddevStorage`	long	storage for data access extension
→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID

→	46	`buffer`	long	pointer to data buffer
→	70	`timeout`	long	timeout value
↔	74	`dataType`	long	data type
↔	82	`len`	word	length of data item
↔	84	`places`	word	decimal places in data item
←	86	`flags`	word	flags

The `DBGetItem` function retrieves the next data item from the data source. See “Retrieving Results” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	No error; no next data item
`rcDBValue`	–801	A nonzero data item was successfully retrieved
`rcDBNull`	–800	The data item was NULL
`rcDBError`	–802	No next data item; execution ended in an error
`rcDBBadType`	–803	Next data item not of requested data type
`rcDBBreak`	–805	Timed out

`DBUnGetItem`

Parameter block

→	00	`message`	word	routine selector; `kDBUnGetItem`
↔	02	`ddevStorage`	long	storage for data access extension
→	06	`asyncPB`	long	pointer to asynch parameter block
→	10	`sessID`	long	session ID

The `DBUnGetItem` function reverses the effect of the last call to the `DBGetItem` function. See “Retrieving Results” in the Data Access Manager chapter of *Inside Macintosh* Volume VI for a complete description of this function.

Result codes

`noErr`	0	No error
`rcDBError`	–802	Error executing function

Further Reference:

- *Inside Macintosh*, Volume VI, Data Access Manager chapter