



PUBLIC

SQL Anywhere - MobiLink

Document Version: 17.01.0 – 2021-10-15

MobiLink - Dbmlsync C++ API Reference

Content

- 1 Dbmsync C++ API Reference 3**
- 1.1 DbmsyncClient Class. 4
 - CancelSync Method. 7
 - Connect(const char *, unsigned, const char *, const char *) Method. 9
 - Disconnect(void) Method. 11
 - Fini(void) Method. 11
 - FreeEventInfo(DBSC_Event *) Method. 12
 - GetErrorInfo(void) Method. 13
 - GetEvent(DBSC_Event **, unsigned) Method. 13
 - GetProperty(const char *, char *) Method. 15
 - Init(void) Method. 15
 - InstantiateClient(void) Method. 16
 - Ping(unsigned) Method. 17
 - SetProperty(const char *, const char *) Method. 17
 - ShutdownServer(DBSC_ShutdownType) Method. 19
 - StartServer(unsigned, const char *, unsigned, DBSC_StartType *) Method. 20
 - Sync(const char *, const char *) Method. 21
 - WaitForServerShutdown(unsigned) Method. 22
- 1.2 DBSC_CancelRet Enumeration. 23
- 1.3 DBSC_ErrorType Enumeration. 24
- 1.4 DBSC_EventType Enumeration. 26
- 1.5 DBSC_GetEventRet Enumeration. 28
- 1.6 DBSC_ShutdownType Enumeration. 29
- 1.7 DBSC_StartType Enumeration. 30
- 1.8 DBSC_ErrorInfo Structure. 30
- 1.9 DBSC_Event Structure. 32

1 Dbmsync C++ API Reference

The Dbmsync C++ API provides a programming interface that allows MobiLink client applications written in C++ to launch synchronizations and receive feedback about the progress of the synchronizations they request.

Header File

```
dbmsynccli.hpp
```

The Dbmsync C++ API reference is available in the *MobiLink - Dbmsync C++ API Reference* at <https://help.sap.com/viewer/38569764c6d64775b4251fe1b3faa435/LATEST/en-US>.

Example

The sample below shows a typical application using the C++ version of the Dbmsync API to perform a synchronization to receive output events. The sample omits error handling for clarity. It is always good practice to check the return value from each API call.

```
#include <stdio.h>
#include "dbmsynccli.hpp"
int main( void ) {
    DbmsyncClient *client;
    DBSC_SyncHdl   syncHdl;
    DBSC_Event     *ev1;
    client = DbmsyncClient::InstantiateClient();
    if( client == NULL ) return( 1 );
    client->Init();
    // Setting the server path is usually not required unless
    // your SQL Anywhere install is not in your path or you have multiple
    // versions of the product installed.
    client->SetProperty( "server path", "C:\\SQLAnywhere\\bin32" );
    client->StartServer( 3426,
        "-c server=remote;dbn=rem1;uid=dba;pwd=passwd -v+ -ot c:\\
        \\dbsync1.txt",
        5000, NULL );
    client->Connect( NULL, 3426, "dba", "sql");
    syncHdl = client->Sync( "my_sync_profile", "" );
    while( client->GetEvent( &ev1, 5000 ) == DBSC_GETEVENT_OK ) {
        if( ev1->hdl == syncHdl ) {
            //
            // Process events that interest you here
            //
            if( ev1->type == DBSC_EVENTTYPE_SYNC_DONE ) {
                client->FreeEventInfo( ev1 );
                break;
            }
            client->FreeEventInfo( ev1 );
        }
    }
    client->ShutdownServer( DBSC_SHUTDOWN_ON_EMPTY_QUEUE );
}
```

```
client->WaitForServerShutdown( 10000 );
client->Disconnect();
client->Fini();
delete client;
return( 0 );
}
```

In this section:

[DbmlsyncClient Class \[page 4\]](#)

Communicates using TCP/IP with a separate process, dbmlsync server, which performs a synchronization by connecting to the MobiLink server and the remote database.

[DBSC_CancelRet Enumeration \[page 23\]](#)

Indicates the result of a synchronization cancellation attempt.

[DBSC_ErrorType Enumeration \[page 24\]](#)

Indicates the reason for a method call failure.

[DBSC_EventType Enumeration \[page 26\]](#)

Indicates the type of event generated by a synchronization.

[DBSC_GetEventRet Enumeration \[page 28\]](#)

Indicates the result of an attempt to retrieve an event.

[DBSC_ShutdownType Enumeration \[page 29\]](#)

Indicates how urgently the server should be shut down.

[DBSC_StartType Enumeration \[page 30\]](#)

Indicates the action taken during a dbmlsync server startup attempt.

[DBSC_ErrorInfo Structure \[page 30\]](#)

Contains information about the failure of a previous method call.

[DBSC_Event Structure \[page 32\]](#)

Contains information about an event generated by a synchronization.

1.1 DbmlsyncClient Class

Communicates using TCP/IP with a separate process, dbmlsync server, which performs a synchronization by connecting to the MobiLink server and the remote database.

≡ Syntax

```
public class DbmlsyncClient
```

Members

All members of DbmlsyncClient, including inherited members.

Methods

Modifier and Type	Method	Description
public virtual bool	CancelSync [page 7]	Cancels a synchronization request.
public virtual bool	Connect(const char *, unsigned, const char *, const char *) [page 9]	Opens a connection to a dbmsync server that is already running on this computer.
public virtual bool	Disconnect(void) [page 11]	Breaks the dbmsync server connection that was established with the Connect method.
public virtual bool	Fini(void) [page 11]	Frees all resources used by this class instance.
public virtual bool	FreeEventInfo(DBSC_Event *) [page 12]	Frees memory associated with a DBSC_Event structure returned by the GetEvent method.
public virtual const DBSC_ErrorInfo *	GetErrorInfo(void) [page 13]	Retrieves additional information about the failure after a DbmsyncClient class method returns a failed return code.
public virtual DBSC_GetEventRet	GetEvent(DBSC_Event **, unsigned) [page 13]	Retrieves the next feedback event for synchronizations requested by the client.
public virtual bool	GetProperty(const char *, char *) [page 15]	Retrieves the current value of a property.
public virtual bool	Init(void) [page 15]	Initializes a DbmsyncClient class instance.
public static DbmsyncClient *	InstantiateClient(void) [page 16]	Creates an instance of the dbmsync client class that can be used to control synchronizations.
public virtual bool	Ping(unsigned) [page 17]	Sends a ping request to the dbmsync server to check if the server is active and responding to requests.
public virtual bool	SetProperty(const char *, const char *) [page 17]	Sets various properties to modify the behavior of the class instance.
public virtual bool	ShutdownServer(DBSC_ShutdownType) [page 19]	Shuts down the dbmsync server to which the client is connected.
public virtual bool	StartServer(unsigned, const char *, unsigned, DBSC_StartType *) [page 20]	Starts a new dbmsync server if one is not already listening on the specified port.
public virtual DBSC_SyncHdl	Sync(const char *, const char *) [page 21]	Requests that the dbmsync server perform a synchronization.
public virtual bool	WaitForServerShutdown(unsigned) [page 22]	Returns when the server has shutdown or when the timeout expires, whichever comes first.

Remarks

Multiple clients can share the same dbmlsync server. However, each dbmlsync server can only synchronize a single remote database. Each remote database can have only one dbmlsync server synchronizing it.

The dbmlsync server performs one synchronization at a time. If the server receives a synchronization request while performing a synchronization, it queues that request and satisfies it later.

Status information generated by synchronizations is communicated back to the client application through the `GetEvent` method.

In this section:

[CancelSync Method \[page 7\]](#)

Cancels a synchronization request.

[Connect\(const char *, unsigned, const char *, const char *\) Method \[page 9\]](#)

Opens a connection to a dbmlsync server that is already running on this computer.

[Disconnect\(void\) Method \[page 11\]](#)

Breaks the dbmlsync server connection that was established with the `Connect` method.

[Fini\(void\) Method \[page 11\]](#)

Frees all resources used by this class instance.

[FreeEventInfo\(DBSC_Event *\) Method \[page 12\]](#)

Frees memory associated with a `DBSC_Event` structure returned by the `GetEvent` method.

[GetErrorInfo\(void\) Method \[page 13\]](#)

Retrieves additional information about the failure after a `DbmlsyncClient` class method returns a failed return code.

[GetEvent\(DBSC_Event **, unsigned\) Method \[page 13\]](#)

Retrieves the next feedback event for synchronizations requested by the client.

[GetProperty\(const char *, char *\) Method \[page 15\]](#)

Retrieves the current value of a property.

[Init\(void\) Method \[page 15\]](#)

Initializes a `DbmlsyncClient` class instance.

[InstantiateClient\(void\) Method \[page 16\]](#)

Creates an instance of the dbmlsync client class that can be used to control synchronizations.

[Ping\(unsigned\) Method \[page 17\]](#)

Sends a ping request to the dbmlsync server to check if the server is active and responding to requests.

[SetProperty\(const char *, const char *\) Method \[page 17\]](#)

Sets various properties to modify the behavior of the class instance.

[ShutdownServer\(DBSC_ShutdownType\) Method \[page 19\]](#)

Shuts down the dbmlsync server to which the client is connected.

[StartServer\(unsigned, const char *, unsigned, DBSC_StartType *\) Method \[page 20\]](#)

Starts a new dbmlsync server if one is not already listening on the specified port.

[Sync\(const char *, const char *\) Method \[page 21\]](#)

Requests that the dbmlsync server perform a synchronization.

[WaitForServerShutdown\(unsigned\) Method \[page 22\]](#)

Returns when the server has shutdown or when the timeout expires, whichever comes first.

Related Information

[GetEvent\(DBSC_Event **, unsigned\) Method \[page 13\]](#)

1.1.1 CancelSync Method

Cancels a synchronization request.

Overload list

Modifier and Type	Overload name	Description
public virtual bool	CancelSync(DBSC_SyncHdl) [page 7]	Allows a client to cancel a synchronization request previously made using the Sync method.
public virtual DBSC_CancelRet	CancelSync(DBSC_SyncHdl, bool) [page 8]	Allows a client to cancel a synchronization request previously made using the Sync method.

In this section:

[CancelSync\(DBSC_SyncHdl\) Method \(Deprecated\) \[page 7\]](#)

Allows a client to cancel a synchronization request previously made using the Sync method.

[CancelSync\(DBSC_SyncHdl, bool\) Method \[page 8\]](#)

Allows a client to cancel a synchronization request previously made using the Sync method.

1.1.1.1 CancelSync(DBSC_SyncHdl) Method (Deprecated)

Allows a client to cancel a synchronization request previously made using the Sync method.

≡ Syntax

```
public virtual bool CancelSync (DBSC_SyncHdl hdl)
```

Parameters

hdl The synchronization handle returned by the Sync method when the synchronization was requested.

Returns

True when the synchronization request was successfully canceled; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

Only synchronization requests waiting to be serviced can be canceled. To stop a synchronization that has already begun, use the CancelSync(UInt32, Boolean) method.

A connection must be established to the server before this method can be used. This method cannot be used if the client has disconnected from the server since the Sync method was called.

Related Information

[GetErrorInfo\(void\) Method \[page 13\]](#)

[CancelSync\(DBSC_SyncHdl, bool\) Method \[page 8\]](#)

[ShutdownServer\(DBSC_ShutdownType\) Method \[page 19\]](#)

1.1.1.2 CancelSync(DBSC_SyncHdl, bool) Method

Allows a client to cancel a synchronization request previously made using the Sync method.

≡ Syntax

```
public virtual DBSC_CancelRet CancelSync (
    DBSC_SyncHdl hdl,
    bool cancel_active
)
```

Parameters

hdl The synchronization handle returned by the Sync method when the synchronization was requested.

cancel_active When set to true, the request is canceled even if the synchronization has already begun. When set to false, the request is only canceled if synchronization has not begun.

Returns

A value from the DBSC_CancelRet enumeration. When DBSC_CANCEL_FAILED is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

A connection must be established to the server before this method can be used. This method cannot be used if the client has disconnected from the server since the Sync method was called.

Related Information

[DBSC_CancelRet Enumeration \[page 23\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

[ShutdownServer\(DBSC_ShutdownType\) Method \[page 19\]](#)

1.1.2 Connect(const char *, unsigned, const char *, const char *) Method

Opens a connection to a dbmsync server that is already running on this computer.

Syntax

```
public virtual bool Connect (
    const char * host,
    unsigned port,
    const char * uid,
    const char * pwd
)
```

Parameters

host This value is reserved. Use NULL.

port The TCP port on which the dbmsync server is listening. Use the same port value that you specified with the StartServer method.

uid A valid database user id with DBA or REMOTE DBA authority on the remote database that is to be synchronized.

pwd The database password for the user specified by uid.

Returns

True when a connection to the server was established; otherwise, returns false. When false is returned, you can call the `GetErrorInfo` method for more information about the failure.

Remarks

Dbmlsync servers are started (either using the command line or the `StartServer` method) using a connection string that provided a database userid and password (for example, *server_userid*). In addition, the `Connect` method of the Dbmlsync API requires a valid database userid (for example, *client_userid*).

Client_userid is used only to validate whether this client has sufficient permissions to synchronize the database. When synchronizations are performed *server_userid* is used.

In SQL Anywhere 12 and earlier, both *client_userid* and *server_userid* required DBA or REMOTE DBA permissions.

In SQL Anywhere 16 and later, *server_userid* must have sufficient privileges to synchronize. At a minimum, *server_userid* must have the SYS_RUN_REPLICATION_ROLE system role but other privileges may be required to allow synchronization. *Client_userid* must either:

- Be the same as *server_userid*, or
- Have the SYS_AUTH_DBA_ROLE system role, or
- Have a user-extended role based on *server_userid*, for example: `CREATE ROLE FOR USER server_userid; GRANT server_userid to client_userid;`

While the last option ensures that *client_userid* has at least as many system privileges as *server_userid*, it is also possible for *server_userid* to have object-level privileges not granted to its role and therefore *client_userid* would not have those privileges. If those privileges are used during synchronization, then *client_userid* has effectively increased its privileges to perform the synchronization. If this is not acceptable, ensure that all of *server_userid*'s object-level privileges are granted to its user-extended role.

Related Information

[StartServer\(unsigned, const char *, unsigned, DBSC_StartType *\) Method \[page 20\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.3 Disconnect(void) Method

Breaks the dbmsync server connection that was established with the Connect method.

☰ Syntax

```
public virtual bool Disconnect (void)
```

Returns

True when the connection to the server has been broken; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

You should always call Disconnect when you are finished with a connection.

Related Information

[Connect\(const char *, unsigned, const char *, const char *\) Method \[page 9\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.4 Fini(void) Method

Frees all resources used by this class instance.

☰ Syntax

```
public virtual bool Fini (void)
```

Returns

True when the class instance is successfully finalized; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

This method must be called before you can delete the DbmlSyncClient class instance.

i Note

You should use the Disconnect method to disconnect from any connected servers before finalizing the class instance.

Related Information

[Disconnect\(void\) Method \[page 11\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.5 FreeEventInfo(DBSC_Event *) Method

Frees memory associated with a DBSC_Event structure returned by the GetEvent method.

≡ Syntax

```
public virtual bool FreeEventInfo (DBSC_Event * event)
```

Parameters

event A pointer to the DBSC_Event structure to be freed.

Returns

True when the memory was successfully freed; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

FreeEventInfo must be called on each DBSC_Event structure returned by the GetEvent method.

Related Information

[DBSC_Event Structure \[page 32\]](#)

[GetEvent\(DBSC_Event **, unsigned\) Method \[page 13\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.6 GetErrorInfo(void) Method

Retrieves additional information about the failure after a DbmlsyncClient class method returns a failed return code.

≡ Syntax

```
public virtual const DBSC_ErrorInfo * GetErrorInfo (void)
```

Returns

A pointer to a DBSC_ErrorInfo structure that contains information about the failure. The contents of this structure may be overwritten the next time any class method is called.

Related Information

[DBSC_ErrorType Enumeration \[page 24\]](#)

[DBSC_ErrorInfo Structure \[page 30\]](#)

1.1.7 GetEvent(DBSC_Event **, unsigned) Method

Retrieves the next feedback event for synchronizations requested by the client.

≡ Syntax

```
public virtual DBSC_GetEventRet GetEvent (
    DBSC_Event ** event,
    unsigned timeout
)
```

Parameters

event If the return value is DBSC_GETEVENT_OK then the event parameter is filled in with a pointer to a DBSC_Event structure containing information about the event that has been retrieved. When you are finished with the event structure you must call the FreeEventInfo method to free memory associated with it.

timeout Indicates the maximum time in milliseconds to wait if no event is immediately available to return. Use DBSC_INFINITY to wait indefinitely for a response.

Returns

A value from the DBSC_GetEventRet enumeration. When DBSC_GETEVENT_FAILED is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

Feedback events contain information such as messages generated from the sync, data for updating a progress bar, and synchronization cycle notifications.

As the dbmsync server runs a synchronization it generates a series of events that contain information about the progress of the synchronization. These events are sent from the server to the DbmsyncClient class, which queues them. When the GetEvent method is called, the next event in the queue is returned if there is one waiting.

If there are no events waiting in the queue, this method waits until an event is available or until the specified timeout has expired before returning.

The types of events that are generated for a synchronization can be controlled using properties.

Related Information

[DBSC_GetEventRet Enumeration \[page 28\]](#)

[DBSC_Event Structure \[page 32\]](#)

[FreeEventInfo\(DBSC_Event *\) Method \[page 12\]](#)

[SetProperty\(const char *, const char *\) Method \[page 17\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.8 GetProperty(const char *, char *) Method

Retrieves the current value of a property.

≡ Syntax

```
public virtual bool GetProperty (  
    const char * name,  
    char * value  
)
```

Parameters

name The name of the property to retrieve. For a list of valid property names, see SetProperty.

value A buffer of at least DBSC_MAX_PROPERTY_LEN bytes where the value of the property is stored.

Returns

True when the property was successfully received; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Related Information

[SetProperty\(const char *, const char *\) Method \[page 17\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.9 Init(void) Method

Initializes a DbmlsyncClient class instance.

≡ Syntax

```
public virtual bool Init (void)
```

Returns

True when the class instance is successfully initialized; otherwise, returns false. When false is returned, you can call the `GetErrorInfo` method for more information about the failure.

Remarks

This method must be called after instantiating the `DbmlSyncClient` class instance. Other `DbmlSyncClient` methods cannot be called until you have successfully initialized the instance.

Related Information

[InstantiateClient\(void\) Method \[page 16\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.10 InstantiateClient(void) Method

Creates an instance of the `dbmlsync` client class that can be used to control synchronizations.

≡ Syntax

```
public static DbmlsyncClient * InstantiateClient (void)
```

Returns

A pointer to the new instance that has been created. Returns null when an error occurs.

Remarks

The pointer returned by this method can be used to call the remaining methods in the class. You can destroy the instance by calling the standard delete operator on the pointer.

1.1.11 Ping(unsigned) Method

Sends a ping request to the dbmlsync server to check if the server is active and responding to requests.

≡ Syntax

```
public virtual bool Ping (unsigned timeout)
```

Parameters

timeout The maximum number of milliseconds to wait for the server to respond to the ping request. Use DBSC_INFINITY to wait indefinitely for a response.

Returns

True when a response to the ping request was received from the server; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

You must be connected to the server before calling this method.

Related Information

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.12 SetProperty(const char *, const char *) Method

Sets various properties to modify the behavior of the class instance.

≡ Syntax

```
public virtual bool SetProperty (  
    const char * name,  
    const char * value  
)
```

Parameters

name The name of the property to set. For a list of valid property names, see table.

value The value to set for the property. The string specified must contain less than DBCS_MAX_PROPERTY_LEN bytes.

Returns

True when the property was successfully set; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

Changes to property values only affect synchronization requests made after the property value was changed.

The *server path* property can be set to specify the directory from which the client should start dbmsync.exe when the StartServer method is called. When this property is not set, dbmsync.exe is found using the PATH environment variable. If there are multiple versions of SQL Anywhere installed on your computer, it is recommended that you specify the location of dbmsync.exe using the *server path* property because the PATH environment variable may locate a dbmsync executable from another installed version of SQL Anywhere. For example,

```
ret = cli->SetProperty("server path", "c:\\sa17\\bin32");
```

The properties control the types of events that are returned by the GetEvent method. By disabling events that you do not require you may be able to improve performance. An event type is enabled by setting the corresponding property to "1" and disabled by setting the property to "0".

The following is a table of available property names and the event types that each name controls:

Property name	Event types controlled	Default value
enable errors	DBSC_EVENTTYPE_ERROR_MSG	1
enable warnings	DBSC_EVENTTYPE_WARNING_MSG	1
enable info msgs	DBSC_EVENTTYPE_INFO_MSG	1
enable progress	DBSC_EVENTTYPE_PROGRESS_IN- DEX	0
enable progress text	DBSC_EVENTTYPE_PROGRESS_TEXT	0
enable title	DBSC_EVENTTYPE_TITLE	0
enable sync start and done	DBSC_EVENTTYPE_SYNC_START DBSC_EVENTTYPE_SYNC_DONE	1

Property name	Event types controlled	Default value
enable status	DBSC_EVENTTYPE_ML_CONNECT DBSC_EVENTTYPE_UPLOAD_COMMITTED DBSC_EVENTTYPE_DOWNLOAD_COMMITTED	1

Related Information

[StartServer\(unsigned, const char *, unsigned, DBSC_StartType *\) Method \[page 20\]](#)

[GetEvent\(DBSC_Event **, unsigned\) Method \[page 13\]](#)

[GetProperty\(const char *, char *\) Method \[page 15\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.13 ShutdownServer(DBSC_ShutdownType) Method

Shuts down the dbmlsync server to which the client is connected.

≡ Syntax

```
public virtual bool ShutdownServer (DBSC_ShutdownType how)
```

Parameters

how Indicates the urgency of the server shutdown. Supported values are listed in the DBSC_ShutdownType enumeration.

Returns

True when a shutdown request was successfully sent to the server; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

The Shutdown method returns immediately but there may be some delay before the server actually shuts down.

The WaitForServerShutdown method can be used to wait until the server actually shuts down.

i Note

You should still use the Disconnect method after calling ShutdownServer.

Related Information

[DBSC_ShutdownType Enumeration \[page 29\]](#)

[Disconnect\(void\) Method \[page 11\]](#)

[WaitForServerShutdown\(unsigned\) Method \[page 22\]](#)

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.14 StartServer(unsigned, const char *, unsigned, DBSC_StartType *) Method

Starts a new dbmsync server if one is not already listening on the specified port.

☰ Syntax

```
public virtual bool StartServer (
    unsigned port,
    const char * cmdline,
    unsigned timeout,
    DBSC_StartType * starttype
)
```

Parameters

port The TCP port to check for an existing dbmsync server. If a new server is started, it is set to listen on this port.

cmdline A valid command line for starting a dbmsync server. The command line may contain only the following options which have the same meaning that they do for the dbmsync utility: -a, -c, -dl, -do, -ek, -ep, -k, -l, -o, -os, -ot, -p, -pc+, -pc-, -pd, -pp, -q, -qi, -qc, -sc, -sp, -uc, -ud, -ui, -um, -un, -ux, -v[chnprsut], -wc, -wh. The -c option must be specified.

timeout The maximum time in milliseconds to wait after a dbmsync server is started for it to be ready to accept requests. Use DBSC_INFINITY to wait indefinitely for a response.

starttype An out parameter set to indicate if the server has been located or started. If starttype is non-null on entry and StartServer returns true, then, on exit, the variable pointed to by starttype is set to a value from the DBSC_StartType enumeration.

Returns

True when the server was already running or successfully started; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

If a server is present, this method sets the starttype parameter to DBSC_SS_ALREADY_RUNNING and returns without further action. If no server is found, the method starts a new server using the options specified by the cmdline argument and waits for it to start accepting requests before returning.

On Windows Mobile devices, it is usually necessary to set the *server path* property before StartServer can be successfully called. The *server path* property does not need to be set in the following instances:

- Your application is in the same directory as dbmlsync.exe.
- dbmlsync.exe is in the Windows directory.

Related Information

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.15 Sync(const char *, const char *) Method

Requests that the dbmlsync server perform a synchronization.

⌵ Syntax

```
public virtual DBSC_SyncHdl Sync (  
    const char * profile_name,  
    const char * extra_opts  
)
```

Parameters

profile_name The name of a synchronization profile defined in the remote database that contains the options for the synchronization. If profile_name is null then no profile is used and the extra_opts parameter should contain all the options for the synchronization.

extra_opts A string formed according to the same rules used to define an option string for a synchronization profile, which is a string specified as a semicolon delimited list of elements of the form <option name>=<option value>. If profile_name is non-null then the options specified by extra_opts are added to those already in the synchronization profile specified by profile_name. If an option in the string already exists in the profile, then the value from the string replaces the value already stored in the profile. If profile_name is null then extra_opts should specify all the options for the synchronization.

Returns

A DBSC_SynchHdl value which uniquely identifies this synchronization request and is only valid until the client disconnects from the server. Returns NULL_SYNCHDL if an error prevents the synchronization request from being created. When NULL_SYNCHDL is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

You must be connected to the server before calling this method. At least one of profile_name and extra_opts must be non-null.

The return value identifies the synchronization request and can be used to cancel the request or to process events returned by the synchronization

Related Information

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.1.16 WaitForServerShutdown(unsigned) Method

Returns when the server has shutdown or when the timeout expires, whichever comes first.

⌘ Syntax

```
public virtual bool WaitForServerShutdown (unsigned timeout)
```

Parameters

timeout Indicates the maximum time in milliseconds to wait for the server to shutdown. Use DBSC_INFINITY to wait indefinitely for a response.

Returns

True when the method returned due to the server shutdown; otherwise, returns false. When false is returned, you can call the GetErrorInfo method for more information about the failure.

Remarks

WaitForServerShutdown can only be called after the ShutdownServer method is called.

Related Information

[GetErrorInfo\(void\) Method \[page 13\]](#)

1.2 DBSC_CancelRet Enumeration

Indicates the result of a synchronization cancellation attempt.

≡ Syntax

```
enum DBSC_CancelRet
```

Members

Member name	Description	Value
DBSC_CANCEL_OK_QUEUED	Canceled a synchronization that was in the wait queue.	1
DBSC_CANCEL_OK_ACTIVE	Canceled an active synchronization.	2
DBSC_CANCEL_FAILED	Failed to cancel the synchronization.	3

Related Information

[CancelSync\(DBSC_SyncHdl\) Method \(Deprecated\) \[page 7\]](#)

1.3 DBSC_ErrorType Enumeration

Indicates the reason for a method call failure.

↳ Syntax

```
enum DBSC_ErrorType
```

Members

Member name	Description	Value
DBSC_ERR_OK	No error occurred.	1
DBSC_ERR_NOT_INITIALIZED	The class has not been initialized by calling the Init method.	2
DBSC_ERR_ALREADY_INITIALIZED	The Init method was called on a class that was already initialized.	3
DBSC_ERR_NOT_CONNECTED	No connection to a dbmsync server is in place.	4
DBSC_ERR_CANT_RESOLVE_HOST	Cannot resolve host information.	5
DBSC_ERR_CONNECT_FAILED	Connection to the dbmsync server has failed.	6
DBSC_ERR_INITIALIZING_TCP_LAYER	Error initializing TCP layer.	7
DBSC_ERR_ALREADY_CONNECTED	Connect method failed because a connection was already in place.	8
DBSC_ERR_PROTOCOL_ERROR	This is an internal error.	9
DBSC_ERR_CONNECTION_REJECTED	The connection was rejected by the dbmsync server. str1 points to a string returned by the server which may provide more information about why the connection attempt was rejected.	10
DBSC_ERR_TIMED_OUT	The timeout expired while waiting for a response from the server.	11

Member name	Description	Value
DBSC_ERR_STILL_CONNECTED	Could not Fini the class because it is still connected to the server.	12
DBSC_ERR_SYNC_NOT_CANCELED	The server could not cancel the synchronization request, likely because the synchronization was already in progress.	14
DBSC_ERR_INVALID_VALUE	An invalid property value was passed to the SetProperty method.	15
DBSC_ERR_INVALID_PROP_NAME	The specified property name is not valid.	16
DBSC_ERR_VALUE_TOO_LONG	The property value is too long; properties must be less than DBCS_MAX_PROPERTY_LEN bytes long.	17
DBSC_ERR_SERVER_SIDE_ERROR	A server-side error occurred while canceling or adding a sync. str1 points to a string returned by the server which may provide more information about the error.	18
DBSC_ERR_CREATE_PROCESS_FAILED	Unable to start a new dbmsync server.	20
DBSC_ERR_READ_FAILED	TCP error occurred while reading data from the dbmsync server.	21
DBSC_ERR_WRITE_FAILED	TCP error occurred while sending data to the dbmsync server.	22
DBSC_ERR_NO_SERVER_RESPONSE	Failed to receive a response from the server that is required to complete the requested action.	23
DBSC_ERR_UID_OR_PWD_TOO_LONG	The UID or PWD specified is too long.	24
DBSC_ERR_UID_OR_PWD_NOT_VALID	The UID or PWD specified is not valid or lacks sufficient privileges to connect.	25
DBSC_ERR_INVALID_PARAMETER	One of the parameters passed to the function was not valid.	26
DBSC_ERR_WAIT_FAILED	An error occurred while waiting for the server to shutdown.	27
DBSC_ERR_SHUTDOWN_NOT_CALLED	WaitForServerShutdown method was called without first calling the ShutdownServer method.	28

Member name	Description	Value
DBSC_ERR_NO_SYNC_ACK	A synchronization request was sent to the server but no acknowledgement was received; There is no way to know if the server received the request. hdl1 is the handle for the sync request that was sent. If the server received the request, this handle can be used to identify events for the synchronization retrieved using the GetEvent method.	29
DBSC_ERR_ACTIVE_SYNC_NOT_CANCELLED	The server could not cancel the synchronization request because the synchronization was active.	30
DBSC_ERR_DEAD_SERVER	The dbmsync server has encountered an error while starting up. The server is now shutting down. Use the dbmsync -o option to log the error message to a file.	31

1.4 DBSC_EventType Enumeration

Indicates the type of event generated by a synchronization.

Syntax

```
enum DBSC_EventType
```

Members

Member name	Description	Value
DBSC_EVENTTYPE_ERROR_MSG	An error was generated by the synchronization; str1 points to the text of the error.	1
DBSC_EVENTTYPE_WARNING_MSG	A warning was generated by the synchronization; str1 points to the text of the warning.	2
DBSC_EVENTTYPE_INFO_MSG	An information message was generated by the synchronization; str1 points to the text of the message.	3

Member name	Description	Value
DBSC_EVENTTYPE_PROGRESS_IN- DEX	Provides information for updating a progress bar; val1 contains the new progress value. This value can range from 0 to 1000, where 0 indicates 0% done and 1000 indicates 100% done.	4
DBSC_EVENTTYPE_PROGRESS_TEXT	The text associated with the progress bar has been updated; the new value is pointed to by str1.	6
DBSC_EVENTTYPE_TITLE	The title for the synchronization window/control has changed; the new title is pointed to by str1.	7
DBSC_EVENTTYPE_SYNC_START	The synchronization has begun; there is no additional information associated with this event.	8
DBSC_EVENTTYPE_SYNC_DONE	The synchronization is complete; val1 contains the exit code from the synchronization. A 0 value indicates success. A non-zero value indicates that the synchronization failed.	9
DBSC_EVENTTYPE_ML_CONNECT	A connection to the MobiLink Server was established; str1 indicates the communication protocol being used and str2 contains the network protocol options used.	10
DBSC_EVENTTYPE_UPLOAD_COM- MITTED	The MobiLink server confirmed that it successfully committed the upload to the consolidated database.	11
DBSC_EVENTTYPE_DOWN- LOAD_COMMITTED	The download has been successfully committed in the remote database. val1 contains the number of insert/update operations committed. val2 contains the number of delete operations committed	12
DBSC_EVENTTYPE_UPLOAD_START	The remote has begun to send an upload to the server.	13

Member name	Description	Value
DBSC_EVENTTYPE_UPLOAD_SENT	The remote has completed sending an upload segment to the server. For incremental and transactional uploads a separate event is generated each time an upload segment is sent. val1 contains the number of insert operations sent. val2 contains the number of update operations sent. val3 contains the number of delete operations sent.	14
DBSC_EVENTTYPE_DOWNLOAD_START	The remote has begun processing the download received from the server.	15

Related Information

[DBSC_Event Structure \[page 32\]](#)

[GetEvent\(DBSC_Event **, unsigned\) Method \[page 13\]](#)

1.5 DBSC_GetEventRet Enumeration

Indicates the result of an attempt to retrieve an event.

☰, Syntax

```
enum DBSC_GetEventRet
```

Members

Member name	Description	Value
DBSC_GETEVENT_OK	Indicates that an event was successfully retrieved.	1
DBSC_GETEVENT_TIMED_OUT	Indicates that the timeout expired without any event being available to return.	2
DBSC_GETEVENT_FAILED	Indicates that no event was returned because of an error condition.	3

Related Information

[GetEvent\(DBSC_Event **, unsigned\) Method \[page 13\]](#)

1.6 DBSC_ShutdownType Enumeration

Indicates how urgently the server should be shut down.

Syntax

```
enum DBSC_ShutdownType
```

Members

Member name	Description	Value
DBSC_SHUT-DOWN_ON_EMPTY_QUEUE	Indicates that the server should complete any outstanding synchronization requests and then shutdown. Once the server receives the shutdown request, it does not accept any more synchronization requests.	1
DBSC_SHUTDOWN_CLEANLY	Indicates that the server should shut-down cleanly, as quickly as possible. If there are outstanding synchronization requests, they are not performed and if there is a running synchronization it may be interrupted.	2

Related Information

[ShutdownServer\(DBSC_ShutdownType\) Method \[page 19\]](#)

1.7 DBSC_StartType Enumeration

Indicates the action taken during a dbmlsync server startup attempt.

Syntax

```
enum DBSC_StartType
```

Members

Member name	Description	Value
DBSC_SS_STARTED	Indicates that a new dbmlsync server was started.	1
DBSC_SS_ALREADY_RUNNING	Indicates that an existing dbmlsync server was found, so no new server was started.	2

Related Information

[StartServer\(unsigned, const char *, unsigned, DBSC_StartType *\) Method \[page 20\]](#)

1.8 DBSC_ErrorInfo Structure

Contains information about the failure of a previous method call.

Syntax

```
typedef struct DBSC_ErrorInfo
```

Members

All members of DBSC_ErrorInfo, including inherited members.

Variables

Modifier and Type	Variable	Description
public DBSC_ErrorType	type	Contains a value that indicates the reason for failure. Supported values are listed in the DBSC_ErrorType enumeration.
public const char *	str1	Contains additional information about the failure. The meaning of this information depends on the value of the type variable.
public const char *	str2	Contains additional information about the failure. The meaning of this information depends on the value of the type variable.
public long int	val1	Contains additional information about the failure. The meaning of this information depends on the value of the type variable.
public long int	val2	Contains additional information about the failure. The meaning of this information depends on the value of the type variable.
public DBSC_SyncHdl	hdl1	Contains additional information about the failure. The meaning of this information depends on the value of the type variable.

Remarks

str1, str2, val1, val2 and hdl1 contain additional information about the failure, and their meanings depend on the error type. The following error types use fields in this structure to store additional information:

- DBSC_ERR_CONNECTION_REJECTED
- DBSC_ERR_SERVER_SIDE_ERROR
- DBSC_ERR_NO_SYNC_ACK

Related Information

[DBSC_ErrorType Enumeration \[page 24\]](#)

1.9 DBSC_Event Structure

Contains information about an event generated by a synchronization.

≡ Syntax

```
typedef struct DBSC_Event
```

Members

All members of DBSC_Event, including inherited members.

Variables

Modifier and Type	Variable	Description
public DBSC_SynchHdl	hdl	Indicates the synchronization that generated the event. This value matches the value returned by the Sync method.
public DBSC_EventType	type	Indicates the type of event being reported.
public const char *	str1	Contains additional information about the event. The meaning of this information depends on the value of the type variable.
public const char *	str2	Contains additional information about the event. The meaning of this information depends on the value of the type variable.
public long int	val1	Contains additional information about the event. The meaning of this information depends on the value of the type variable.
public long int	val2	Contains additional information about the event. The meaning of this information depends on the value of the type variable.

Modifier and Type	Variable	Description
public long int	val3	Contains additional information about the event. The meaning of this information depends on the value of the type variable.
public void *	data	Contains additional information about the event. The meaning of this information depends on the value of the type variable.

Related Information



[DBSC_EventType Enumeration \[page 26\]](#)

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.

About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
 - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
 - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.

The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <https://www.sap.com/about/legal/trademark.html> for additional trademark information and notices.