PUBLIC
SQL Anywhere - UltraLite
Document Version: 17.01.0 – 2021-10-15

UltraLite - Java API Reference

THE BEST RUN SAP

Content

1 UltraLiteJ API Reference

UltraLiteJ has a variety of API objects.

The following list describes some of the commonly used API objects:

DatabaseManager

Provides methods for managing databases and connections.

Connection

Represents a connection to an UltraLite database. You can create one or more Connection objects.

SyncParms

Synchronizes your UltraLite database with a MobiLink server.

PreparedStatement, ResultSet

Create dynamic SQL statements, make queries, execute INSERT, UPDATE, and DELETE statements, and attain programmatic control over database result sets.

Package [Android]

```
com.sap.ultralitejni17
```

In this section:

DecimalNumber Interface [page 67]
  Describes an exact decimal value and provides decimal arithmetic support for Java platforms where java.math.BigDecimal is not available.

Domain Interface [page 72]
  Describes the Domain object type information for a column in a table.

FileTransfer Interface [page 76]
  Provides a mechanism to transfer files between the client and a MobiLink server.

FileTransferProgressData Interface [page 100]
  Reports file transfer progress monitoring data.

FileTransferProgressListener Interface [page 102]
  Receives file transfer progress events.

IndexSchema Interface [page 104]
  Specifies the schema of an index and provides constants that are useful for querying system tables.

PreparedStatement Interface [page 105]
  Provides methods to execute a SQL query to generate a ResultSet object or to execute a prepared SQL statement on a database.

ResultSet Interface [page 129]
  Provides methods to traverse a table by row, and access the column data.

ResultSetMetadata Interface [page 158]
  Associated with a ResultSet object and contains a method that provides column information.

SQLInfo Interface [page 167]
  Represents information about an executed SQL statement.

StreamHTTPParms Interface [page 170]
  Represents HTTP stream parameters that define how to communicate with a MobiLink server using HTTP.

StreamHTTPSParms Interface [page 183]
  Represents HTTPS stream parameters that define how to communicate with a MobiLink server using secure HTTPS connections.

StreamTCPIPParms Interface [page 190]
  Represents TCP/IP stream parameters that define how to communicate with a MobiLink server using TCP/IP.

StreamTLSParms Interface [page 191]
  Represents TLS stream parameters that define how to communicate with a MobiLink server using secure TLS connections.

SyncObserver Interface [page 193]
  Receives synchronization progress information.

SyncObserver.States Interface [page 195]
  Defines the synchronization states that can be signaled to an observer.

SyncParms Class [page 197]
  Maintains the parameters used during the database synchronization process.

SyncResult Class [page 224]
  Reports status-related information about a specified database synchronization.

SyncResult.AuthStatusCode Interface [page 238]
    Enumerates the authorization codes returned by the MobiLink server.

TableSchema Interface [page 239]
    Specifies the schema of a table and provides constants defining the names of system tables.

ULjEvent Interface [page 241]
    Represents an UltraLiteJ API system event.

ULjException Class [page 243]
    Supersedes the exceptions thrown by the database.

Unsigned64 Class [page 246]
    Implements unsigned 64-bit binary values.

UUIDValue Interface [page 253]
    Describes a unique identifier (UUID or Universally Unique IDentifier) object.

ValidateDatabaseProgressData Interface [page 255]
    Reports ValidateDatabase progress data.

ValidateDatabaseProgressData.StatusId Interface [page 257]
    Specifies possible status IDs for the UltraLite Validate Database utility.

ValidateDatabaseProgressListener Interface [page 259]
    Receives ValidateDatabase progress events.

1.1 ColumnSchema Interface

Specifies the schema of a column.

Syntax

```
public interface ColumnSchema
```

Members

All members of ColumnSchema, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final byte	COLUMN_DEFAULT_NONE	Indicates that the column has no default attribute. The following defaults apply when no default attribute is assigned: • Nullable columns default to null • Not nullable numeric columns default to zero • Not nullable varying length columns default to zero length values. The default value of existing tables can be determined by querying the column_default column in the TableSchema.SYS_COLUMNS system table.
public static final byte	COLUMN_DEFAULT_AUTOINC	Indicates the AUTOINCREMENT column default attribute. When using the AUTOINCREMENT attribute, the column must be one of the integer data types, or an exact numeric type. On an INSERT, if a value is not specified for the AUTOINCREMENT column, a unique value larger than any other value in the column is generated. If an INSERT specifies a value for the column that is larger than the current maximum value for the column, the value is used as a starting point for subsequent inserts. In UltraLiteJ, the autoincremented value is not set to 0 when the table is created, and the AUTOINCREMENT attribute generates negative numbers when a signed data type is used for the column. Therefore, declare AUTOINCREMENT columns as unsigned integers to prevent negative values from being used. The default value of existing tables can be determined by querying the column_default column in the TableSchema.SYS_COLUMNS system table.

Modifier and Type	Variable	Description
public static final byte	COLUMN_DEFAULT_GLOBAL_AUTO-INC	Indicates the GLOBAL AUTOINCRE-MENT column default attribute. This constant is similar to the AUTO-INCREMENT attribute, but the domain is partitioned. Each partition contains the same number of values. You must assign each copy of the database a unique global database identification number. UltraLiteJ supplies default val-ues in a database from the partition uniquely identified by that database's number. The default value of existing tables can be determined by querying the col-umn_default column in the Table-Schema.SYS_COLUMNS system table.
public static final byte	COLUMN_DEFAULT_CURRENT_DATE	Indicates the CURRENT DATE (year, month, and day) column default attrib-ute. See "CURRENT DATE special value" un-der "Special values in UltraLite" in the SQL Anywhere documentation set. The default value of existing tables can be determined by querying the col-umn_default column in the Table-Schema.SYS_COLUMNS system table.
public static final byte	COLUMN_DEFAULT_CURRENT_TIME	Indicates the CURRENT TIME column default attribute. See "CURRENT TIME special value" un-der "Special values in UltraLite" in the SQL Anywhere documentation set. The default value of existing tables can be determined by querying the col-umn_default column in the Table-Schema.SYS_COLUMNS system table.

Modifier and Type	Variable	Description
public static final byte	COLUMN_DEFAULT_CURRENT_TIME-STAMP	Indicates the CURRENT TIMESTAMP column default attribute. This constant combines the CURRENT DATE and CURRENT TIME values to form a TIMESTAMP value, which contains the year, month, day, hour, minute, second, and fraction of a second. The precision of the fraction is set to 3 decimal places. The accuracy of this constant is limited by the accuracy of the system clock. See "CURRENT TIMESTAMP special value" under "Special values in UltraLite" in the SQL Anywhere documentation set. The default value of existing tables can be determined by querying the column_default column in the TableSchema.SYS_COLUMNS system table.
public static final byte	COLUMN_DEFAULT_UNIQUE_ID	Indicates a new unique identifier column default attribute. UUIDs can be used to uniquely identify rows in a table. The generated values are unique on every computer or device, meaning they can be used as keys in synchronization and replication environments. The default value of existing tables can be determined by querying the column_default column in the TableSchema.SYS_COLUMNS system table.
public static final byte	COLUMN_DEFAULT_CONSTANT	Indicates a constant column default attribute. The default value of existing tables can be determined by querying the column_default column in the TableSchema.SYS_COLUMNS system table.

Modifier and Type	Variable	Description
public static final byte	COLUMN_DEFAULT_CUR-RENT_UTC_TIMESTAMP	Indicates the CURRENT UTC TIME-STAMP column default attribute. This constant combines the CURRENT DATE and CURRENT TIME values to form a UTC TIMESTAMP value, which contains the year, month, day, hour, mi-nute, second, and fraction of a second as observed in GMT. The precision of the fraction is set to 3 decimal places. The accuracy of this constant is limited by the accuracy of the system clock. See "CURRENT UTC TIMESTAMP spe-cial value" under "Special values in Ul-traLite" in the SQL Anywhere documen-tation set. The default value of existing tables can be determined by querying the col-umn_default column in the Table-Schema.SYS_COLUMNS system table.
public static final byte	COLUMN_DEFAULT_AUTOFILENAME	Indicates the AUTOFILENAME column default attribute. When a VARCHAR column has this de-fault value, the column is the filename column in an external blob definition. When a column has this type of default, the column_default_value column in the TableSchema.SYS_COLUMNS system table contains the prefix and extension strings found in the external blob defini-tion, in the form 'prefix|extension'. The default value of existing tables can be determined by querying the col-umn_default column in the Table-Schema.SYS_COLUMNS system table.

Remarks

This interface only contains constants for different column default values stored in column_default column of the syscolumn system table.

1.2 ConfigFile Interface

Establishes a Configuration object for a persistent database saved in a file.

Syntax

```
public interface ConfigFile extends ConfigPersistent
```

Members

All members of ConfigFile, including inherited members.

Inherited members from ConfigPersistent

Modifier and Type	Member	Description
public void	enableAesDBEncryption() [page 21]	Enables AES encryption of the data-base.
public void	enableObfuscation() [page 22]	Enables obfuscation of the database.
public int	getCacheSize() [page 22]	Returns the cache size of the database, in bytes.
public String	getConnectionString() [page 23]	Gets the connection string registered with the setConnectionString method.
public String	getCreationString() [page 23]	Gets the creation string registered with the setCreationString method.
public String	getEncryptionKey() [page 24]	Gets the database encryption key that was registered with the setEncryption-Key method.
public String	getUserName() [page 24]	Gets the name of user set by the setU-serName method.
public ConfigPersistent	setCacheSize(int) [page 25]	Sets the cache size of the database, in bytes.
public void	setConnectionString(String) [page 25]	Sets the connection string to be used to create or connect to a database.
public void	setCreationString(String) [page 26]	Sets the creation string to be used to create a database.
public void	setEncryptionKey(String) [page 26]	Sets the key for encryption.
public void	setUserName(String) [page 27]	Sets the name of the user.

Inherited members from Configuration

Modifier and Type	Member	Description
public String	getDatabaseName() [page 28]	Returns the database name.

Modifier and Type	Member	Description
public int	getPageSize() [page 29]	Returns the page size of the database, in bytes.
public Configuration	setDatabaseName(String) [page 29]	Sets the database name.
public Configuration	setPageSize(int) [page 30]	Sets the page size of the database.
public Configuration	setPassword(String) [page 30]	Sets the database password.

Related Information

DatabaseManager Class [page 62]

1.3 ConfigFileAndroid Interface

Establishes a Configuration object for a persistent database saved in a file.

Syntax

```
public interface ConfigFileAndroid extends ConfigFile
```

Members

All members of ConfigFileAndroid, including inherited members.

Inherited members from ConfigPersistent

Modifier and Type	Member	Description
public void	enableAesDBEncryption() [page 21]	Enables AES encryption of the data-base.
public void	enableObfuscation() [page 22]	Enables obfuscation of the database.
public int	getCacheSize() [page 22]	Returns the cache size of the database, in bytes.
public String	getConnectionString() [page 23]	Gets the connection string registered with the setConnectionString method.
public String	getCreationString() [page 23]	Gets the creation string registered with the setCreationString method.

Modifier and Type	Member	Description
public String	getEncryptionKey() [page 24]	Gets the database encryption key that was registered with the setEncryption-Key method.
public String	getUserName() [page 24]	Gets the name of user set by the setU-serName method.
public ConfigPersistent	setCacheSize(int) [page 25]	Sets the cache size of the database, in bytes.
public void	setConnectionString(String) [page 25]	Sets the connection string to be used to create or connect to a database.
public void	setCreationString(String) [page 26]	Sets the creation string to be used to create a database.
public void	setEncryptionKey(String) [page 26]	Sets the key for encryption.
public void	setUserName(String) [page 27]	Sets the name of the user.

Inherited members from Configuration

Modifier and Type	Member	Description
public String	getDatabaseName() [page 28]	Returns the database name.
public int	getPageSize() [page 29]	Returns the page size of the database, in bytes.
public Configuration	setDatabaseName(String) [page 29]	Sets the database name.
public Configuration	setPageSize(int) [page 30]	Sets the page size of the database.
public Configuration	setPassword(String) [page 30]	Sets the database password.

Remarks

An object implementing the ConfigFileAndroid interface is created by using the DatabaseManager.createConfigurationFileAndroid method.

Related Information

DatabaseManager Class [page 62]
createConfigurationFileAndroid(String, android.content.Context) Method [page 64]

1.4 ConfigPersistent Interface

Establishes a Configuration object for a persistent database.

Syntax

```
public interface ConfigPersistent extends Configuration
```

Members

All members of ConfigPersistent, including inherited members.

Methods

Modifier and Type	Method	Description
public void	enableAesDBEncryption() [page 21]	Enables AES encryption of the data-base.
public void	enableObfuscation() [page 22]	Enables obfuscation of the database.
public int	getCacheSize() [page 22]	Returns the cache size of the database, in bytes.
public String	getConnectionString() [page 23]	Gets the connection string registered with the setConnectionString method.
public String	getCreationString() [page 23]	Gets the creation string registered with the setCreationString method.
public String	getEncryptionKey() [page 24]	Gets the database encryption key that was registered with the setEncryption-Key method.
public String	getUserName() [page 24]	Gets the name of user set by the setU-serName method.
public ConfigPersistent	setCacheSize(int) [page 25]	Sets the cache size of the database, in bytes.
public void	setConnectionString(String) [page 25]	Sets the connection string to be used to create or connect to a database.
public void	setCreationString(String) [page 26]	Sets the creation string to be used to create a database.
public void	setEncryptionKey(String) [page 26]	Sets the key for encryption.
public void	setUserName(String) [page 27]	Sets the name of the user.

Inherited members from Configuration

Modifier and Type	Member	Description
public String	getDatabaseName() [page 28]	Returns the database name.

Modifier and Type	Member	Description
public int	getPageSize() [page 29]	Returns the page size of the database, in bytes.
public Configuration	setDatabaseName(String) [page 29]	Sets the database name.
public Configuration	setPageSize(int) [page 30]	Sets the page size of the database.
public Configuration	setPassword(String) [page 30]	Sets the database password.

In this section:

1.4.1 enableAesDBEncryption() Method

Enables AES encryption of the database.

Syntax

```
public void enableAesDBEncryption ()
```

Remarks

Specify the DBKEY connection parameter when creating or connecting to the database, or use the setEncryptionKey method.

Related Information

setEncryptionKey(String) Method [page 26]

1.4.2 enableObfuscation() Method

Enables obfuscation of the database.

Syntax

```
public void enableObfuscation ()
```

1.4.3 getCacheSize() Method

Returns the cache size of the database, in bytes.

Syntax

```
public int getCacheSize ()
```

Returns

The cache size.

Related Information

setCacheSize(int) Method [page 25]

1.4.4 getConnectionString() Method

Gets the connection string registered with the setConnectionString method.

Syntax

```
public String getConnectionString ()
```

Returns

The connection string registered with the setConnectionString method.

Related Information

setConnectionString(String) Method [page 25]

1.4.5 getCreationString() Method

Gets the creation string registered with the setCreationString method.

Syntax

```
public String getCreationString ()
```

Returns

The creation string registered with the setCreationString method.

Related Information

setCreationString(String) Method [page 26]

1.4.6 getEncryptionKey() Method

Gets the database encryption key that was registered with the setEncryptionKey method.

Syntax

```
public String getEncryptionKey ()
```

Returns

The database encryption key that was registered with the setEncryptionKey method.

Related Information

setEncryptionKey(String) Method [page 26]
changeEncryptionKey(String) Method [page 39]

1.4.7 getUserName() Method

Gets the name of user set by the setUserName method.

Syntax

```
public String getUserName ()
```

Returns

The name of user set by the setUserName method.

Related Information

setUserName(String) Method [page 27]

1.4.8 setCacheSize(int) Method

Sets the cache size of the database, in bytes.

Syntax

```
public ConfigPersistent setCacheSize (int cache_size) throws ULjException
```

Parameters

cache_size The cache size. The default cache size is 20480 (20KB) on all platforms.

Returns

This ConfigPersistent object with the cache size specified.

Remarks

The cache size determines the number of database pages resident in the page cache. Increasing the size means less reading and writing of database pages, at the expense of increased time to locate pages in the cache.

Related Information

getCacheSize() Method [page 22]

1.4.9 setConnectionString(String) Method

Sets the connection string to be used to create or connect to a database.

Syntax

```
public void setConnectionString (String connection_string)
```

Parameters

connection_string The connection string used in database connection or creation.

Remarks

Any other items that have been set in this configuration are also passed to create or connect to a database.

1.4.10 setCreationString(String) Method

Sets the creation string to be used to create a database.

Syntax

```
public void setCreationString (String creation_string)
```

Parameters

creation_string The creation string used in database creation.

Remarks

Any other items that have been set in this configuration are also passed to create a database.

1.4.11 setEncryptionKey(String) Method

Sets the key for encryption.

Syntax

```
public void setEncryptionKey (String encryption_key)
```

Parameters

encryption_key The string to use for the encryption key.

Related Information

getEncryptionKey() Method [page 24]
enableAesDBEncryption() Method [page 21]
changeEncryptionKey(String) Method [page 39]

1.4.12 setUserName(String) Method

Sets the name of the user.

Syntax

```
public void setUserName (String user_name)
```

Parameters

user_name The name of the user.

Remarks

This name is used to connect or create to the native database with the UID= phrase in the connection string.

1.5 Configuration Interface

Establishes a Configuration object for a database.

Syntax

```
public interface Configuration
```

Members

All members of Configuration, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getDatabaseName() [page 28]	Returns the database name.
public int	getPageSize() [page 29]	Returns the page size of the database, in bytes.
public Configuration	setDatabaseName(String) [page 29]	Sets the database name.
public Configuration	setPageSize(int) [page 30]	Sets the page size of the database.
public Configuration	setPassword(String) [page 30]	Sets the database password.

Remarks

Some attributes are used only during database creation while others apply to the initial connection to a database. Attributes are ignored if they are set after creating a database, or connecting to a database.

In this section:

1.5.1 getDatabaseName() Method

Returns the database name.

Syntax

```
public String getDatabaseName ()
```

Returns

The name of the database.

1.5.2 getPageSize() Method

Returns the page size of the database, in bytes.

Syntax

```
public int getPageSize ()
```

Returns

The page size.

1.5.3 setDatabaseName(String) Method

Sets the database name.

Syntax

```
public Configuration setDatabaseName (String db_name) throws ULjException
```

Parameters

db_name The name of database.

Returns

This Configuration object with the database name specified.

1.5.4 setPageSize(int) Method

Sets the page size of the database.

Syntax

```
public Configuration setPageSize (int page_size) throws ULjException
```

Parameters

page_size The page size, in bytes.

Returns

This Configuration object with the page size specified.

Remarks

The page size setting is used to determine the maximum size of a row stored in a persistent database. It establishes the size of an index page, and determines the number of children that each page can have.

When using an existing database, the size is already set to the page size of the database when it was created. You can not reset the page size of an existing database using this method.

The page size can be 1024, 2048, 4096, 8192, or 16384 bytes. The default is 4096 bytes.

1.5.5 setPassword(String) Method

Sets the database password.

Syntax

```
public Configuration setPassword (String password) throws ULjException
```

Parameters

password A password for a new database, or the password to gain access to an existing database.

Returns

This Configuration object with the database password set.

Remarks

The password is used to gain access to the database, and must match the password specified when the database was created. The default is "dba".

1.6 Connection Interface

Describes a database connection, which is required to initiate database operations.

Syntax

```
public interface Connection
```

Members

All members of Connection, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final byte	CONNECTED	Denotes a connected state.
public static final byte	NOT_CONNECTED	Denotes a not connected state.
public static final String	OPTION_DATABASE_ID	Database option: database id. A default value is not specified. It must be explicitly assigned.
public static final String	OPTION_DATE_FORMAT	Database option: date format. Set this option in the database creation string of the ConfigPersistent.setCreationString method only. The default value of the corresponding option is "YYYY-MM-DD".

Modifier and Type	Variable	Description
public static final String	OPTION_DATE_ORDER	Database option: date order. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "YMD".
public static final String	OPTION_MAX_HASH_SIZE	Database option: maximum hash size. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The option name is max_hash_size. The default value of the corresponding option is "4". If the SQL statement creating an index does not specify a hash size, the value specified by this option is used as the default.
public static final String	OPTION_ML_REMOTE_ID	Database option: ML remote ID. A default value is not specified. A value is set after the first MobiLink synchroni-zation.
public static final String	OPTION_NEAREST_CENTURY	Database option: nearest century. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "50".
public static final String	OPTION_PRECISION	Database option: precision. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "30".
public static final String	OPTION_SCALE	Database option: scale. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "6".

Modifier and Type	Variable	Description
public static final String	OPTION_TIME_FORMAT	Database option: time format. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "HH:NN:SS.SSS".
public static final String	OPTION_TIMESTAMP_FORMAT	Database option: timestamp format. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "YYYY-MM-DD HH:NN:SS.SSS".
public static final String	OPTION_TIMESTAMP_INCREMENT	Database option: timestamp increment. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "1".
public static final String	OPTION_TIME-STAMP_WITH_TIME_ZONE_FORMAT	Database option: timestamp with time zone format. Set this option in the database creation string with the ConfigPersistent.set-CreationString method only. The default value of the corresponding option is "YYYY-MM-DD HH:NN:SS.SSS +HH:NN".
public static final String	PROPERTY_DATABASE_NAME	Database Property: database name. Set this property with the Configura-tion.setDatabaseName method.
public static final String	PROPERTY_PAGE_SIZE	Database Property: page size. Set this property with the Configura-tion.setPageSize method.

Modifier and Type	Variable	Description
public static final String	SYNC_ALL	The publication list used to request synchronization of all tables in the database, including tables not in any publication. Tables marked as NoSync are never synchronized. This constant is equivalent to the null reference or an empty string.
public static final String	SYNC_ALL_DB_PUB_NAME	The reserved name of the SYNC_ALL_DB publication.
public static final String	SYNC_ALL_PUBS	The publication list used to request synchronization of all publications in the database. Tables that are marked as NoSync are never synchronized.
public static final int	ULVF_DATABASE	Used to validate database. Verify all database pages using page checksums and additional checks.
public static final int	ULVF_EXPRESS	Used to perform a faster, though less thorough, validation. This flag modifies others specified.
public static final int	ULVF_FULL_VALIDATE	Performs all types of validation on the database.
public static final int	ULVF_INDEX	Used to validate indexes. Check the integrity of the index.
public static final int	ULVF_TABLE	Used to validate table(s). Check that table and index row counts match.

Methods

Modifier and Type	Method	Description
public void	cancelWaitForEvent() [page 38]	Cancels any waitForEvent calls on this Connection object.
public void	changeEncryptionKey(String) [page 39]	Changes the database encryption key for an UltraLite database.
public void	commit() [page 39]	Commits the database changes.
public DecimalNumber	createDecimalNumber [page 40]	Creates a new DecimalNumber object.
public SyncParms	createSyncParms [page 42]	Creates a set of synchronization parameters.

Modifier and Type	Method	Description
public UUIDValue	createUUIDValue() [page 44]	Creates a UUID value.
public void	dropDatabase() [page 44]	Drops a database.
public DatabaseInfo	getDatabaseInfo() [page 44]	Returns a DataInfo object containing information about database properties.
public String	getDatabaseProperty(String) [page 45]	Returns a database property.
public Date	getLastDownloadTime(String) [page 45]	Returns the time of the most recent download of the specified publication.
public long	getLastIdentity() [page 46]	Retrieves the most recent value inserted into a DEFAULT AUTOINCREMENT or DEFAULT GLOBAL AUTOINCREMENT column, or zero if the most recent INSERT transaction was made on a table that had no such column.
public SQLInfo	getLastWarning() [page 46]	Returns information about the last SQL statement executed on this connection.
public String	getOption(String) [page 47]	Returns a database option.
public byte	getState() [page 47]	Returns the state of the connection.
public SyncObserver	getSyncObserver() [page 48]	Returns the SyncObserver object that is currently registered for this Connection object.
public SyncResult	getSyncResult() [page 49]	Returns the result of the last SYNCHRONIZE SQL statement.
public PreparedStatement	prepareStatement(String) [page 50]	Prepares a statement for execution.
public void	registerForEvent(short, String) [page 50]	Registers a system event to receive notifications.
public void	release() [page 51]	Releases this connection.
public void	resetLastDownloadTime(String) [page 51]	Resets the time of the download for the specified publications.
public void	rollback() [page 52]	Commits a rollback to undo changes to the database.
public void	rollbackPartialDownload() [page 52]	Rolls back the changes from a failed synchronization.
public void	setDatabaseId(int) [page 53]	Sets the database ID for global autoincrement.
public void	setOption(String, String) [page 53]	Sets the database option.
public void	setSyncObserver(SyncObserver) [page 54]	Sets a SyncObserver object to monitor the progress of synchronizations on this connection.
public void	synchronize(SyncParms) [page 54]	Synchronizes the database with a MobiLink server.

Modifier and Type	Method	Description
public void	unregisterForEvent(short, String) [page 55]	Unregisters from a system event to stop receiving notifications.
public void	validateDatabase(int, ValidateDatabaseProgressListener, String) [page 55]	Validates the database on this connection.
public ULjEvent	waitForEvent(int) [page 56]	Waits for an event notification.

Remarks

A connection is obtained using the connect or createDatabase methods of the DatabaseManager class. Use the release method when the connection is no longer needed. When all connections for a database are released, the database is closed.

A Connection object provides the following capabilities:

- Create new schema (tables, indexes and publications)
- Create new value and domain objects
- Permanently commit changes to the database
- Prepare SQL statements for execution
- Roll back uncommitted changes to the database

The following example demonstrates how to create a schema for a simple database with a Connection object, conn, created for it. The database contains a table named T1, which has a single integer primary key column named num, and a table named T2, which has an integer primary key column named num and an integer column named quantity. T2 has an addition index on quantity. A publication named PubA contains T1.

```
// Assumes a valid connection object, conn, for the current database.
PreparedStatement ps;
ps = conn.prepareStatement( "CREATE TABLE T1 ( num INT NOT NULL PRIMARY KEY )" );
ps.execute();
ps.close();
ps = conn.prepareStatement( "CREATE TABLE T2 ( num INT NOT NULL PRIMARY KEY,
quantity INT)" );
ps.execute();
ps.close();
ps = conn.prepareStatement( "CREATE INDEX index1 ON T2( quantity )" );
ps.execute();
ps.close();
ps = conn.prepareStatement( "CREATE Publication PubA ( Table T1 )" );
ps.execute();
ps.close();
```

In this section:

createDecimalNumber Method [page 40]
: Creates a new DecimalNumber object.

createSyncParms Method [page 42]
: Creates a set of synchronization parameters.

createUUIDValue() Method [page 44]
: Creates a UUID value.

dropDatabase() Method [page 44]
: Drops a database.

getDatabaseInfo() Method [page 44]
: Returns a DataInfo object containing information about database properties.

getDatabaseProperty(String) Method [page 45]
: Returns a database property.

getLastDownloadTime(String) Method [page 45]
: Returns the time of the most recent download of the specified publication.

getLastIdentity() Method [page 46]
: Retrieves the most recent value inserted into a DEFAULT AUTOINCREMENT or DEFAULT GLOBAL AUTOINCREMENT column, or zero if the most recent INSERT transaction was made on a table that had no such column.

getLastWarning() Method [page 46]
: Returns information about the last SQL statement executed on this connection.

getOption(String) Method [page 47]
: Returns a database option.

getState() Method [page 47]
: Returns the state of the connection.

getSyncObserver() Method [page 48]
: Returns the SyncObserver object that is currently registered for this Connection object.

getSyncResult() Method [page 49]
: Returns the result of the last SYNCHRONIZE SQL statement.

prepareStatement(String) Method [page 50]
: Prepares a statement for execution.

registerForEvent(short, String) Method [page 50]
: Registers a system event to receive notifications.

release() Method [page 51]
: Releases this connection.

resetLastDownloadTime(String) Method [page 51]
: Resets the time of the download for the specified publications.

rollback() Method [page 52]
: Commits a rollback to undo changes to the database.

rollbackPartialDownload() Method [page 52]
: Rolls back the changes from a failed synchronization.

setDatabaseId(int) Method [page 53]
: Sets the database ID for global autoincrement.

setOption(String, String) Method [page 53]
    Sets the database option.

setSyncObserver(SyncObserver) Method [page 54]
    Sets a SyncObserver object to monitor the progress of synchronizations on this connection.

synchronize(SyncParms) Method [page 54]
    Synchronizes the database with a MobiLink server.

unregisterForEvent(short, String) Method [page 55]
    Unregisters from a system event to stop receiving notifications.

validateDatabase(int, ValidateDatabaseProgressListener, String) Method [page 55]
    Validates the database on this connection.

waitForEvent(int) Method [page 56]
    Waits for an event notification.

Related Information

DatabaseManager Class [page 62]
createDatabase(Configuration) Method [page 65]
connect(Configuration) Method [page 63]
release() Method [page 51]

1.6.1 cancelWaitForEvent() Method

Cancels any waitForEvent calls on this Connection object.

Syntax

```
public void cancelWaitForEvent () throws ULjException
```

Related Information

waitForEvent(int) Method [page 56]

1.6.2 changeEncryptionKey(String) Method

Changes the database encryption key for an UltraLite database.

Syntax

```
public void changeEncryptionKey (String newKey) throws ULjException
```

Parameters

newKey The new encryption key for the database.

Remarks

Applications that call this method must first ensure that the user has either synchronized the database or created a reliable backup copy of the database. It is important to have a reliable backup of the database because the changeEncryptionKey method is an operation that must run to completion. When the database encryption key is changed, every row in the database is first decrypted with the old key and then encrypted with the new key and rewritten. This operation is not recoverable. If the encryption change operation does not complete, the database is left in an invalid state and you cannot access it again.

1.6.3 commit() Method

Commits the database changes.

Syntax

```
public void commit () throws ULjException
```

Remarks

Invoking this method causes all table data changes since the last commit or rollback to become permanent.

1.6.4 createDecimalNumber Method

Creates a new DecimalNumber object.

Overload list

Modifier and Type	Overload name	Description
public DecimalNumber	createDecimalNumber(int, int) [page 40]	Creates a DecimalNumber object.
public DecimalNumber	createDecimalNumber(int, int, String) [page 41]	Creates a DecimalNumber object.

In this section:

createDecimalNumber(int, int) Method [page 40]
    Creates a DecimalNumber object.

createDecimalNumber(int, int, String) Method [page 41]
    Creates a DecimalNumber object.

1.6.4.1 createDecimalNumber(int, int) Method

Creates a DecimalNumber object.

Syntax

```
public DecimalNumber createDecimalNumber (
    int precision,
    int scale
) throws ULjException
```

Parameters

precision The number of digits in the number.
scale The number of decimal places in the number.

Returns

The DecimalNumber object with the specified type.

Related Information

DecimalNumber Interface [page 67]

1.6.4.2 createDecimalNumber(int, int, String) Method

Creates a DecimalNumber object.

Syntax

```
public DecimalNumber createDecimalNumber (
    int precision,
    int scale,
    String value
) throws ULjException
```

Parameters

precision The number of digits in the number.
scale The number of decimal places in the number.
value The value to be set.

Returns

The DecimalNumber object with the specified type.

Related Information

DecimalNumber Interface [page 67]

1.6.5 createSyncParms Method

Creates a set of synchronization parameters.

Overload list

Modifier and Type	Overload name	Description
public SyncParms	createSyncParms(String, String) [page 42]	Creates a set of synchronization parameters for HTTP synchronization.
public SyncParms	createSyncParms(int, String, String) [page 43]	Creates a set of synchronization parameters for HTTP synchronization.

In this section:

1.6.5.1 createSyncParms(String, String) Method

Creates a set of synchronization parameters for HTTP synchronization.

Syntax

```
public SyncParms createSyncParms (
    String userName,
    String version
) throws ULjException
```

Parameters

userName The unique MobiLink user name for this client database.
version The MobiLink script version.

Returns

The SyncParms object.

Related Information

createSyncParms(int, String, String) Method [page 43]
setUserName(String) Method [page 222]

1.6.5.2 createSyncParms(int, String, String) Method

Creates a set of synchronization parameters for HTTP synchronization.

Syntax

```
public SyncParms createSyncParms (
    int streamType,
    String userName,
    String version
) throws ULjException
```

Parameters

streamType One of the constants defined in the SyncParms class used to identify the type of synchronization stream.
userName The MobiLink user name.
version The MobiLink script version.

Returns

A SyncParms object.

Related Information

createSyncParms(String, String) Method [page 42]

1.6.6 createUUIDValue() Method

Creates a UUID value.

Syntax

```
public UUIDValue createUUIDValue () throws ULjException
```

Returns

The UUIDValue instance for the domain.

1.6.7 dropDatabase() Method

Drops a database.

Syntax

```
public void dropDatabase () throws ULjException
```

Remarks

The database referenced by the connection is erased and the connection is released. This connection is the only one that can be active for the database being dropped.

1.6.8 getDatabaseInfo() Method

Returns a DataInfo object containing information about database properties.

Syntax

```
public DatabaseInfo getDatabaseInfo () throws ULjException
```

Returns

The DatabaseInfo object.

1.6.9 getDatabaseProperty(String) Method

Returns a database property.

Syntax

```
public String getDatabaseProperty (String name) throws ULjException
```

Parameters

name The name of the database property. You can set this parameter to any supported UltraLite database property name.

Returns

The value of the property that corresponds to the given name.

1.6.10 getLastDownloadTime(String) Method

Returns the time of the most recent download of the specified publication.

Syntax

```
public Date getLastDownloadTime (String pub_name) throws ULjException
```

Parameters

pub_name The name of the publication to check. This parameter must reference a single publication or be the special Connection.SYNC_ALL_DB_PUB_NAME publication for the time of the last download of the full database.

Returns

The timestamp of the last download.

Related Information

resetLastDownloadTime(String) Method [page 51]

1.6.11 getLastIdentity() Method

Retrieves the most recent value inserted into a DEFAULT AUTOINCREMENT or DEFAULT GLOBAL AUTOINCREMENT column, or zero if the most recent INSERT transaction was made on a table that had no such column.

Syntax

```
public long getLastIdentity ()
```

Returns

The most recent identity value.

Remarks

If a table contains more than one column of the (GLOBAL) AUTOINCREMENT type, then the column to which this value belongs to is undetermined.

1.6.12 getLastWarning() Method

Returns information about the last SQL statement executed on this connection.

Syntax

```
public SQLInfo getLastWarning ()
```

Returns

The SQLInfo object for the last SQL statement executed.

1.6.13 getOption(String) Method

Returns a database option.

Syntax

```
public String getOption (String option_name) throws ULjException
```

Parameters

option_name The name of option to get.

option_name The name of the option to get. You can set this parameter to any constant in the Connection interface that has an *OPTION_* prefix.

Returns

The value of the database option.

Remarks

Database options are stored within the database and may also be obtained when a database is connected at some time after the option has been set.

A set of required options are created when a database is created.

Related Information

setOption(String, String) Method [page 53]

1.6.14 getState() Method

Returns the state of the connection.

Syntax

```
public byte getState () throws ULjException
```

Returns

The byte representing the state of the connection.

Remarks

The following example demonstrates how to check the connection state and release the connection.

```
if( _conn.getState() == Connection.CONNECTED ){
    _conn.release();
}
```

Related Information

Connection Interface [page 31]

1.6.15 getSyncObserver() Method

Returns the SyncObserver object that is currently registered for this Connection object.

Syntax

```
public SyncObserver getSyncObserver ()
```

Returns

The SyncObserver, or null if no observer exists.

Related Information

setSyncObserver(SyncObserver) Method [page 54]

1.6.16 getSyncResult() Method

Returns the result of the last SYNCHRONIZE SQL statement.

Syntax

```
public SyncResult getSyncResult ()
```

Returns

The SyncResult object representing the result of the last SYNCHRONIZE SQL statement.

Remarks

The following example illustrates how to get the result of the last SYNCHRONIZE SQL statement:

```
PreparedStatement ps = conn.prepareStatement("SYNCHRONIZE PROFILE myprofile");
ps.execute();
ps.close();
SyncResult result = conn.getSyncResult();
display(
   "*** Synchronized *** sent=" + result.getSentRowCount()
   + ", received=" + result.getReceivedRowCount()
);
```

i Note

This method does not return the result of the last call to the Connection.synchronize method. To obtain the SyncResult object for the last Connection.synchronize(SyncParms) method call, use the getSyncResult method on the SyncParms object passed in.

Related Information

SyncResult Class [page 224]
getSyncResult() Method [page 208]

1.6.17 prepareStatement(String) Method

Prepares a statement for execution.

Syntax

```
public PreparedStatement prepareStatement (String sql) throws ULjException
```

Parameters

sql A SQL statement to prepare.

Returns

A PreparedStatement object.

Related Information

PreparedStatement Interface [page 105]

1.6.18 registerForEvent(short, String) Method

Registers a system event to receive notifications.

Syntax

```
public void registerForEvent (
    short event_type,
    String object_name
) throws ULjException
```

Parameters

event_type The type of event to register for.
object_name The object to which the event applies, such as a table name.

Related Information

ULjEvent Interface [page 241]

1.6.19 release() Method

Releases this connection.

Syntax

```
public void release () throws ULjException
```

Remarks

Once a connection has been released, it can no longer be used to access the database.

It is an error release a connection for which there exist uncommitted transactions.

1.6.20 resetLastDownloadTime(String) Method

Resets the time of the download for the specified publications.

Syntax

```
public void resetLastDownloadTime (String pub_name) throws ULjException
```

Parameters

pub_name The name of the publication to check.

Remarks

To reset the download time for when the entire database is synchronized, use the special Connection.SYNC_ALL_DB_PUB_NAME publication.

This method requires that no uncommitted transactions are on the current connection.

1.6.21 rollback() Method

Commits a rollback to undo changes to the database.

Syntax

```
public void rollback () throws ULjException
```

Remarks

Invoking this method undoes all table data changes on this Connection object since the last commit or rollback.

1.6.22 rollbackPartialDownload() Method

Rolls back the changes from a failed synchronization.

Syntax

```
public void rollbackPartialDownload () throws ULjException
```

Remarks

This method only affects resumable downloads. (synchronizing with SyncParms.setKeepPartialDownload set to true)

If a communication error occurs during the download phase of synchronization while the KeepPartialDownload parameter is true, the downloaded changes are retained so that synchronization can resume from the place where the download is interrupted.

This method discards the partial download when you no longer want to resume the download.

Related Information

setKeepPartialDownload(boolean) Method [page 214]

1.6.23 setDatabaseId(int) Method

Sets the database ID for global autoincrement.

Syntax

```
public void setDatabaseId (int id) throws ULjException
```

Parameters

id The database ID.

Remarks

The database ID does not have a default value.

During an INSERT, GLOBAL AUTOINCREMENT columns have NULL values inserted unless a database ID has been set explicitly.

1.6.24 setOption(String, String) Method

Sets the database option.

Syntax

```
public void setOption (
    String option_name,
    String option_value
) throws ULjException
```

Parameters

option_name The name of the option to set. You can set this parameter to any supported UltraLite database option name.
option_value The new value of the option.

Remarks

If the option is not currently stored on the database, it is created.

There cannot be any uncommitted transactions for this connection when the method is invoked.

1.6.25 setSyncObserver(SyncObserver) Method

Sets a SyncObserver object to monitor the progress of synchronizations on this connection.

Syntax

```
public void setSyncObserver (SyncObserver so)
```

Parameters

so A SyncObserver object or null to remove the currently registered SyncObserver object.

Remarks

This SyncObserver object is used by subsequent SYNCHRONIZE SQL statements.

The default is null, suggesting no observer.

Related Information

SyncObserver Interface [page 193]

1.6.26 synchronize(SyncParms) Method

Synchronizes the database with a MobiLink server.

Syntax

```
public void synchronize (SyncParms config) throws ULjException
```

Parameters

config The SyncParms object containing the parameters used for synchronization.

Related Information

SyncParms Class [page 197]

1.6.27 unregisterForEvent(short, String) Method

Unregisters from a system event to stop receiving notifications.

Syntax

```
public void unregisterForEvent (
    short event_type,
    String object_name
) throws ULjException
```

Parameters

event_type The type of event to unregister.
object_name The object to which the event applies, such as a table name.

Related Information

ULjEvent Interface [page 241]

1.6.28 validateDatabase(int, ValidateDatabaseProgressListener, String) Method

Validates the database on this connection.

Syntax

```
public void validateDatabase (
    int flags,
    ValidateDatabaseProgressListener listener,
    String tableName
) throws ULjException
```

Parameters

flags Flags controlling the type of validation.
listener The listener to receive validation progress information.
tableName A specific table to validate, or null for all tables.

Remarks

Tables, indexes, and database pages can be validated depending on the flags passed to this routine. To receive information during the validation, implement a callback function and pass the address to this routine. To limit the validation to a specific table, pass in the table name or ID as the last parameter.

The flags parameter is a combination of the following values:

- ULVF_TABLE
- ULVF_INDEX
- ULVF_DATABASE
- ULVF_EXPRESS
- ULVF_FULL_VALIDATE

The following example demonstrates table and index validation in express mode:

```
flags = ULVF_TABLE | ULVF_INDEX | ULVF_EXPRESS;
```

1.6.29 waitForEvent(int) Method

Waits for an event notification.

Syntax

```
public ULjEvent waitForEvent (int wait_ms) throws ULjException
```

Parameters

wait_ms The time, in milliseconds, to wait (block) before returning. To wait indefinitely, set to -1.

Returns

The event that occurred within the wait time, or null if no notification was received within the wait time.

Remarks

This call blocks until a notification is received or until the given wait period expires. To cancel a wait, use the cancelWaitForEvent method.

Related Information

ULjEvent Interface [page 241]
cancelWaitForEvent() Method [page 38]

1.7 DatabaseInfo Interface

Associated with a Connection object and provides methods to reveal database information.

Syntax

```
public interface DatabaseInfo
```

Members

All members of DatabaseInfo, including inherited members.

Methods

Modifier and Type	Method	Description
public int	getNumberRowsToUpload [page 58]	Returns the number of rows awaiting upload.
public int	getPageReads() [page 60]	Returns the number of page reads.
public int	getPageSize() [page 60]	Returns the page size of the database, in bytes.
public int	getPageWrites() [page 61]	Returns the number of page writes.
public String	getRelease() [page 61]	Returns the software release number.

Remarks

This interface is invoked with the getDatabaseInfo method of a Connection object.

In this section:

getNumberRowsToUpload Method [page 58]
    Returns the number of rows awaiting upload.

getPageReads() Method [page 60]
    Returns the number of page reads.

getPageSize() Method [page 60]
    Returns the page size of the database, in bytes.

getPageWrites() Method [page 61]
    Returns the number of page writes.

getRelease() Method [page 61]
    Returns the software release number.

Related Information

Connection Interface [page 31]
getDatabaseInfo() Method [page 44]

1.7.1 getNumberRowsToUpload Method

Returns the number of rows awaiting upload.

Overload list

Modifier and Type	Overload name	Description
public int	getNumberRowsToUpload() [page 59]	Returns the number of rows awaiting upload.
public int	getNumberRowsToUpload(String, int) [page 59]	Returns the number of rows awaiting upload up to a given threshold.

In this section:

getNumberRowsToUpload() Method [page 59]
    Returns the number of rows awaiting upload.

getNumberRowsToUpload(String, int) Method [page 59]

Returns the number of rows awaiting upload up to a given threshold.

1.7.1.1 getNumberRowsToUpload() Method

Returns the number of rows awaiting upload.

Syntax

```
public int getNumberRowsToUpload ()
```

Returns

The number of rows.

1.7.1.2 getNumberRowsToUpload(String, int) Method

Returns the number of rows awaiting upload up to a given threshold.

Syntax

```
public int getNumberRowsToUpload (
    String publList,
    int threshold
)
```

Parameters

pubList A string containing a comma-separated list of publications to check. An empty string (the UL_SYNC_ALL macro) implies all tables except tables marked as *no sync*. A string containing just an asterisk (the UL_SYNC_ALL_PUBS macro) implies that all tables referred to in any publication. Some tables may not be part of any publication and are not included if this value is *.

threshold The maximum number of rows to count, limiting the amount of time taken by the call. A threshold of 0 corresponds to no limit (all rows that need to be synchronized are counted). A threshold of 1 can be used to quickly determine if any rows need to be synchronized.

Returns

The number of rows that need to be synchronized, either in a specified set of publications or in the whole database.

1.7.2 getPageReads() Method

Returns the number of page reads.

Syntax

```
public int getPageReads ()
```

Returns

The number of page reads.

Remarks

This number is the total accumulated page reads and stored in an instance variable of this class.

1.7.3 getPageSize() Method

Returns the page size of the database, in bytes.

Syntax

```
public int getPageSize ()
```

Returns

The page size.

1.7.4 getPageWrites() Method

Returns the number of page writes.

Syntax

```
public int getPageWrites ()
```

Returns

The number of page writes.

Remarks

This number is the total accumulated page writes and stored in an instance variable of this class.

1.7.5 getRelease() Method

Returns the software release number.

Syntax

```
public String getRelease ()
```

Returns

The release number.

Remarks

For example, a software release value of "12.0.1.1234" represents the 12.0.1 release and the 1234 build number.

1.8 DatabaseManager Class

Provides static methods to obtain basic configurations, create a new database, and connect to an existing database.

Syntax

```
public class DatabaseManager
```

Members

All members of DatabaseManager, including inherited members.

Methods

Modifier and Type	Method	Description
public static Connection	connect(Configuration) [page 63]	Connects to an existing database based on a configuration set.
public static ConfigFileAndroid	createConfigurationFileAndroid(String, android.content.Context) [page 64]	Creates a Configuration object for a physical database store from a file and returns a ConfigFileAndroid object.
public static Connection	createDatabase(Configuration) [page 65]	Creates a new database based on a set of configurations, and connects to the database.
public static FileTransfer	createFileTransfer(String, int, String, String) [page 65]	Creates a FileTransfer object for transferring files to or from MobiLink.
public static FileTransfer	createFileTransferAndroid(android.content.Context, String, int, String, String) [page 66]	Creates a FileTransfer object for transferring files to or from MobiLink.
public static void	release() [page 67]	Closes this DatabaseManager object to release all connections and to shutdown all databases.

Remarks

The following example demonstrates how to open an existing database or create a new one if it does not exist:

```
Connection conn = null;
ConfigFileAndroid config = null;
try {
   config = DatabaseManager.createConfigurationFileAndroid(
        "test.udb", getApplicationContext()
   );
   conn = DatabaseManager.connect(config);
} catch(ULjException ex) {
    if (config != null) {
        try {
            conn = DatabaseManager.createDatabase(config);
            // Create the schema here.
        } catch(ULjException exception) {
            // An error has occurred.
        }
    }
}
```

In this section:

Related Information

Connection Interface [page 31]
Configuration Interface [page 27]

1.8.1 connect(Configuration) Method

Connects to an existing database based on a configuration set.

Syntax

```
public static Connection connect (Configuration config) throws ULjException
```

Parameters

config The Configuration object containing the specifications for the existing database.

Returns

A Connection object that establishes the connection to the database.

Related Information

Configuration Interface [page 27]
Connection Interface [page 31]

1.8.2 createConfigurationFileAndroid(String, android.content.Context) Method

Creates a Configuration object for a physical database store from a file and returns a ConfigFileAndroid object.

Syntax

```
public static ConfigFileAndroid createConfigurationFileAndroid (
    String file_name,
    android.content.Context context
) throws ULjException
```

Parameters

file_name The name of the database file to use or create. You must have read-write access to the database path. The default path for this file is `/data/data/your-application-package-name/`, where *your-application-package-name* is the package name you assigned to your application. You can include an absolute path with the file name to specify a different location for the database.
context The Context object from an Android application. This parameter must not be null.

Returns

The ConfigFileAndroid object used to configure a database.

Related Information

ConfigFileAndroid Interface [page 18]

1.8.3 createDatabase(Configuration) Method

Creates a new database based on a set of configurations, and connects to the database.

Syntax

```
public static Connection createDatabase (Configuration config) throws
ULjException
```

Parameters

config A Configuration object containing the specifications for the new database.

Returns

A Connection object that establishes a connection the new database.

Remarks

This method replaces any existing database on the device that may have the same name.

Related Information

Configuration Interface [page 27]
Connection Interface [page 31]

1.8.4 createFileTransfer(String, int, String, String) Method

Creates a FileTransfer object for transferring files to or from MobiLink.

Syntax

```
public static FileTransfer createFileTransfer (
    String fileName,
    int streamType,
    String userName,
    String version
) throws ULjException
```

Parameters

fileName The name of the server file to transfer. This parameter must not contain any path information.
streamType One of the constants defined in the SyncParms class used to identify the type of communication stream.
userName The MobiLink user name.
version The MobiLink script version.

Returns

The FileTransfer object.

Related Information

SyncParms Class [page 197]

1.8.5 createFileTransferAndroid(android.content.Context, String, int, String, String) Method

Creates a FileTransfer object for transferring files to or from MobiLink.

Syntax

```
public static FileTransfer createFileTransferAndroid (
    android.content.Context context,
    String fileName,
    int streamType,
    String userName,
    String version
) throws ULjException
```

Parameters

context The Context object from an Android application. This parameter must not be null.
fileName The name of the server file to transfer. This parameter must not contain any path information.
streamType One of the constants defined in the SyncParms class used to identify the type of communication stream.
userName The MobiLink user name.
version The MobiLink script version.

Returns

The FileTransfer object.

Remarks

This method must be used if a database connection has not been created using the createConfigurationFileAndroid method.

Related Information

createConfigurationFileAndroid(String, android.content.Context) Method [page 64]

1.8.6 release() Method

Closes this DatabaseManager object to release all connections and to shutdown all databases.

Syntax

```
public static void release () throws ULjException
```

Remarks

This method releases all connections that were created with this DatabaseManager.

Any uncommitted transactions are rolled back.

1.9 DecimalNumber Interface

Describes an exact decimal value and provides decimal arithmetic support for Java platforms where java.math.BigDecimal is not available.

Syntax

```
public interface DecimalNumber
```

Members

All members of DecimalNumber, including inherited members.

Methods

Modifier and Type	Method	Description
public DecimalNumber	add(DecimalNumber, DecimalNumber) [page 69]	Adds two DecimalNumber objects together and returns the sum.
public DecimalNumber	divide(DecimalNumber, DecimalNumber) [page 69]	Divides the first DecimalNumber object by the second DecimalNumber object returns the quotient.
public String	getString() [page 70]	Returns the String representation of the DecimalNumber object.
public boolean	isNull() [page 70]	Determines if the DecimalNumber object is null.
public DecimalNumber	multiply(DecimalNumber, DecimalNumber) [page 70]	Multiplies two DecimalNumber objects together and returns the product.
public void	set(String) [page 71]	Sets the DecimalNumber object with a String value.
public void	setNull() [page 71]	Sets the DecimalNumber object to null.
public DecimalNumber	subtract(DecimalNumber, DecimalNumber) [page 72]	Subtracts the second DecimalNumber object from the first DecimalNumber object and returns the difference.

In this section:

1.9.1 add(DecimalNumber, DecimalNumber) Method

Adds two DecimalNumber objects together and returns the sum.

Syntax

```
public DecimalNumber add (
    DecimalNumber num1,
    DecimalNumber num2
) throws ULjException
```

Parameters

num1 A number.

num2 Another number.

Returns

The sum of num1 and num2.

1.9.2 divide(DecimalNumber, DecimalNumber) Method

Divides the first DecimalNumber object by the second DecimalNumber object returns the quotient.

Syntax

```
public DecimalNumber divide (
    DecimalNumber num1,
    DecimalNumber num2
) throws ULjException
```

Parameters

num1 A dividend.

num2 A divisor.

Returns

The quotient of num1 divided by num2.

1.9.3 getString() Method

Returns the String representation of the DecimalNumber object.

Syntax

```
public String getString () throws ULjException
```

Returns

The String value.

1.9.4 isNull() Method

Determines if the DecimalNumber object is null.

Syntax

```
public boolean isNull ()
```

Returns

True if the object is null; otherwise, returns false.

1.9.5 multiply(DecimalNumber, DecimalNumber) Method

Multiplies two DecimalNumber objects together and returns the product.

Syntax

```
public DecimalNumber multiply (
     DecimalNumber num1,
     DecimalNumber num2
) throws ULjException
```

Parameters

num1 A multiplicand.

num2 A multiplier.

Returns

The product of num1 and num2.

1.9.6 set(String) Method

Sets the DecimalNumber object with a String value.

Syntax

```
public void set (String value) throws ULjException
```

Parameters

value A numerical value represented as a String.

1.9.7 setNull() Method

Sets the DecimalNumber object to null.

Syntax

```
public void setNull () throws ULjException
```

1.9.8 subtract(DecimalNumber, DecimalNumber) Method

Subtracts the second DecimalNumber object from the first DecimalNumber object and returns the difference.

Syntax

```
public DecimalNumber subtract (
    DecimalNumber num1,
    DecimalNumber num2
) throws ULjException
```

Parameters

num1 A minuend.

num2 A subtrahend.

Returns

The difference between num1 and num2.

1.10 Domain Interface

Describes the Domain object type information for a column in a table.

Syntax

```
public interface Domain
```

Members

All members of Domain, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final short	SHORT	Denotes the domain ID constant for a 16-bit integer (SMALLINT SQL type).

Modifier and Type	Variable	Description
public static final short	INTEGER	Denotes the domain ID constant for a 32-bit integer (INTEGER SQL type).
public static final short	NUMERIC	Denotes the domain ID constant for a numeric value of fixed precision (size) total digits and with scale digits after the decimal (NUMERIC(precision,scale) SQL type).
public static final short	REAL	Denotes the domain ID constant for a 4-byte floating point (REAL SQL type).
public static final short	DOUBLE	Denotes the domain ID constant for a 8-byte floating point (DOUBLE SQL type).
public static final short	DATE	Denotes the domain ID constant for a Date (DATE SQL type).
public static final short	VARCHAR	Denotes the domain ID constant for a variable-length character string of maximum size bytes (VARCHAR(size) SQL type).
public static final short	LONGVARCHAR	Denotes the domain ID constant for an arbitrary long block of character data (CLOB) (LONG VARCHAR SQL type).
public static final short	BINARY	Denotes the domain ID constant for a variable-length binary object of maximum size bytes (BINARY(size) SQL type).
public static final short	LONGBINARY	Denotes the domain ID constant for an arbitrary long block of binary data (BLOB) (LONG BINARY SQL type).
public static final short	TIMESTAMP	Denotes the domain ID constant for a Timestamp (TIMESTAMP SQL type).
public static final short	TIME	Denotes the domain ID constant for a Time (TIME SQL type).
public static final short	TINY	Denotes the domain ID constant for a unsigned 8-bit integer (TINYINT SQL type).
public static final short	BIG	Denotes the domain ID constant for a 64-bit integer (BIGINT SQL type).
public static final short	UNSIGNED_INTEGER	Denotes the domain ID constant for a unsigned 32-bit integer (UNSIGNED INTEGER SQL type).
public static final short	UNSIGNED_SHORT	Denotes the domain ID constant for a unsigned 16-bit integer (UNSIGNED SMALLINT SQL type).
public static final short	UNSIGNED_BIG	Denotes the domain ID constant for a unsigned 64-bit integer (UNSIGNED BIGINT SQL type).

Modifier and Type	Variable	Description
public static final short	BIT	Denotes the domain ID constant for a bit (BIT SQL type). BIT columns are not nullable by default.
public static final short	UUID	Denotes the domain ID constant for a UniqueIdentifier (UNIQUEIDENTIFIER SQL type).
public static final short	ST_GEOMETRY	Denotes the denotes the domain ID constant for a geometry (GEOMETRY SQL type)
public static final short	LONGBINARYFILE	Denotes the domain ID constant for an arbitrary file of data.
public static final short	TIMESTAMP_ZONE	Denotes the domain ID constant for a timestamps with time zones (DATETI-MEOFFSET SQL type).
public static final short	DOMAIN_MAX	Denotes the maximum kinds of Domain types.

Remarks

This interface contains constants to denote the various domains, and methods that extract information from a Domain object.

See the Connection interface for an example of creating a schema for a simple database.

Types can be classified as follows:

Integer Types:

Domain Constant	SQL Type	Value Range
BIT	BIT	0 or 1
TINY	TINYINT	0 to 255 (unsigned integer using 1 byte of storage)
SHORT	SMALLINT	-32768 to 32767 (signed integer using 2 bytes of storage)
UNSIGNED_SHORT	UNSIGNED SMALLINT	0 to 65535 (unsigned integer using 2 bytes of storage)
INTEGER	INTEGER	-2^{31} to 2^{31} - 1, or -2147483648 to 2147483647 (signed integer using 4 bytes of storage)
UNSIGNED_INTEGER	UNSIGNED INTEGER	0 to 2^{32} - 1, or 0 to 4294967295 (un-signed integer using 4 bytes of storage)

Domain Constant	SQL Type	Value Range
BIG	BIGINT	-2^{63} to 2^{63} - 1, or -9223372036854775808 to 9223372036854775807 (signed integer using 8 bytes of storage)
UNSIGNED_BIG	UNSIGNED BIGINT	0 to 2^{64} - 1, or 0 to 18446744073709551615 (unsigned in-teger using 8 bytes of storage)

Non-Integer Numeric Types:

Domain Constant	SQL Type	Value Range
REAL	REAL	-3.402823e+38 to 3.402823e+38, with numbers close to zero as small as 1.175495e-38 (single precision floating point number using 4 bytes of storage, rounding errors may occur after the sixth digit)
DOUBLE	DOUBLE	-1.79769313486231e+308 to 1.79769313486231e+308, with num-bers close to zero as small as 2.22507385850721e-308 (single preci-sion floating point number using 8 bytes of storage, rounding errors may occur after the fifteenth digit)
NUMERIC	NUMERIC(precision,scale)	Any decimal numbers with *precision* (size) total digits and with *scale* digits after the decimal point (no rounding within precision)

Character and Binary Types:

Domain Constant	SQL Type	Size Range
VARCHAR	VARCHAR(size)	1 to 32767 bytes (characters are stored as 1-3 byte UTF-8 characters). When evaluating expressions, the maximum length for a temporary character value is 2048 bytes.
LONGVARCHAR	LONG VARCHAR	Any length (memory permitting).The only operations allowed on LONG VAR-CHAR columns are to insert, update, or delete them, or to include them in the select-list of a query.
BINARY	BINARY(size)	1 to 32767 bytes. When evaluating ex-pressions, the maximum length for a temporary character value is 2048 bytes.

Domain Constant	SQL Type	Size Range
LONGBINARY	LONG BINARY	Any length (memory permitting).The only operations allowed on LONG BINARY columns are to insert, update, or delete them, or to include them in the select-list of a query.
UUID	UNIQUEIDENTIFIER	Always 16 bytes binary with special interpretation.

Date and Time Types:

Domain Constant	SQL Type	Value
DATE	DATE	Year, month, day.
TIME	TIME	Hour, minute, second, and fraction of a second.
TIMESTAMP	TIMESTAMP	DATE and TIME.
TIMESTAMP_ZONE	TIMESTAMP_ZONE	DATE and TIME with time zone.

BIT columns are not nullable by default. All other types are nullable by default.

Related Information

Connection Interface [page 31]

1.11 FileTransfer Interface

Provides a mechanism to transfer files between the client and a MobiLink server.

Syntax

```
public interface FileTransfer
```

Members

All members of FileTransfer, including inherited members.

Methods

Modifier and Type	Method	Description
public abstract boolean	downloadFile [page 80]	Downloads the file with the specified properties of this object.
public abstract String	getAuthenticationParms() [page 82]	Returns parameters provided to a custom user authentication script.
public abstract int	getAuthStatus() [page 83]	Returns the authorization status code of the last file transfer attempt.
public abstract long	getAuthValue() [page 83]	Returns the value specified in custom user authentication synchronization scripts.
public abstract int	getFileAuthCode() [page 83]	Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.
public abstract int	getLivenessTimeout() [page 84]	Returns the liveness timeout length, in seconds.
public abstract String	getLocalFileName() [page 84]	Determines the local file name.
public abstract String	getLocalPath() [page 85]	Specifies where to find or store the file in the local file system.
public abstract String	getPassword() [page 85]	Returns the MobiLink password for the user specified with the setUserName method.
public abstract String	getRemoteKey() [page 86]	Determines the current remote key value.
public abstract String	getServerFileName() [page 86]	Returns the name of the file on the server.
public abstract int	getStreamErrorCode() [page 87]	Returns the error code reported by the stream.
public abstract String	getStreamErrorMessage() [page 87]	Returns the error message reported by the stream itself.
public abstract StreamHTTPParms	getStreamParms() [page 88]	Returns the parameters used to configure the synchronization stream.
public abstract String	getUserName() [page 88]	Returns the MobiLink user name that uniquely identifies the client to the MobiLink server.
public abstract String	getVersion() [page 89]	Returns the synchronization script to use.
public abstract boolean	isResumePartialTransfer() [page 89]	Determines whether to resume or discard a previous partial transfer.
public abstract boolean	isTransferredFile() [page 90]	Checks whether the file was actually downloaded during the last file transfer attempt.
public abstract void	setAuthenticationParms(String) [page 90]	Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Modifier and Type	Method	Description
public abstract void	setLivenessTimeout(int) [page 91]	Sets the liveness timeout length, in seconds.
public abstract void	setLocalFileName(String) [page 92]	Specifies the local file name.
public abstract void	setLocalPath(String) [page 92]	Specifies where to find or store the file in the local file system.
public abstract void	setPassword(String) [page 93]	Sets the MobiLink password for the user specified with the setUserName method.
public abstract void	setRemoteKey(String) [page 94]	Specifies the remote key.
public abstract void	setResumePartialTransfer(boolean) [page 95]	Specifies whether to resume or discard a previous partial transfer.
public abstract void	setServerFileName(String) [page 95]	Specifies the name of the file on the server.
public abstract void	setUserName(String) [page 96]	Sets the MobiLink user name that uniquely identifies the client to the MobiLink server.
public abstract void	setVersion(String) [page 97]	Sets the synchronization script to use.
public abstract boolean	uploadFile [page 98]	Uploads the file with the specified properties of this object.

Remarks

A FileTransfer object is obtained by calling the DatabaseManager.createFileTransfer method.

The instance returned by the createFileTransfer method can be used to transfer any files between MobiLink and the local file system.

The local file system is either a media card or the internal file system where the application has appropriate permissions. Eg. /sdcard/Android/data/your.package.name/files/

i Note

The application should not simultaneously start two downloads to the same local file.

In this section:

downloadFile Method [page 80]
Downloads the file with the specified properties of this object.

getAuthenticationParms() Method [page 82]
Returns parameters provided to a custom user authentication script.

getAuthStatus() Method [page 83]
Returns the authorization status code of the last file transfer attempt.

getAuthValue() Method [page 83]

Returns the value specified in custom user authentication synchronization scripts.

getFileAuthCode() Method [page 83]
: Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.

getLivenessTimeout() Method [page 84]
: Returns the liveness timeout length, in seconds.

getLocalFileName() Method [page 84]
: Determines the local file name.

getLocalPath() Method [page 85]
: Specifies where to find or store the file in the local file system.

getPassword() Method [page 85]
: Returns the MobiLink password for the user specified with the setUserName method.

getRemoteKey() Method [page 86]
: Determines the current remote key value.

getServerFileName() Method [page 86]
: Returns the name of the file on the server.

getStreamErrorCode() Method [page 87]
: Returns the error code reported by the stream.

getStreamErrorMessage() Method [page 87]
: Returns the error message reported by the stream itself.

getStreamParms() Method [page 88]
: Returns the parameters used to configure the synchronization stream.

getUserName() Method [page 88]
: Returns the MobiLink user name that uniquely identifies the client to the MobiLink server.

getVersion() Method [page 89]
: Returns the synchronization script to use.

isResumePartialTransfer() Method [page 89]
: Determines whether to resume or discard a previous partial transfer.

isTransferredFile() Method [page 90]
: Checks whether the file was actually downloaded during the last file transfer attempt.

setAuthenticationParms(String) Method [page 90]
: Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

setLivenessTimeout(int) Method [page 91]
: Sets the liveness timeout length, in seconds.

setLocalFileName(String) Method [page 92]
: Specifies the local file name.

setLocalPath(String) Method [page 92]
: Specifies where to find or store the file in the local file system.

setPassword(String) Method [page 93]
: Sets the MobiLink password for the user specified with the setUserName method.

setRemoteKey(String) Method [page 94]
: Specifies the remote key.

setResumePartialTransfer(boolean) Method [page 95]
Specifies whether to resume or discard a previous partial transfer.

setServerFileName(String) Method [page 95]
Specifies the name of the file on the server.

setUserName(String) Method [page 96]
Sets the MobiLink user name that uniquely identifies the client to the MobiLink server.

setVersion(String) Method [page 97]
Sets the synchronization script to use.

uploadFile Method [page 98]
Uploads the file with the specified properties of this object.

Related Information

createFileTransfer(String, int, String, String) Method [page 65]

1.11.1 downloadFile Method

Downloads the file with the specified properties of this object.

Overload list

Modifier and Type	Overload name	Description
public abstract boolean	downloadFile() [page 81]	Downloads the file with the specified properties of this object.
public abstract boolean	downloadFile(FileTransferProgressListener) [page 81]	Download the file specified by the properties of this object with progress events posted to the specified listener.

In this section:

downloadFile() Method [page 81]
Downloads the file with the specified properties of this object.

downloadFile(FileTransferProgressListener) Method [page 81]
Download the file specified by the properties of this object with progress events posted to the specified listener.

1.11.1.1 downloadFile() Method

Downloads the file with the specified properties of this object.

Syntax

```
public abstract boolean downloadFile () throws ULjException
```

Returns

True if download is successful; otherwise, a ULjException is thrown and the method does not return normally.

Remarks

The file specified by the setServerFileName method is downloaded from the MobiLink server to the path specified by the setLocalPath method using the specified stream, userName, password, and script version.

Further options can be specified using the setLocalFileName(), setAuthenticationParms() and setResumePartialTransfer() methods.

A detailed result status can be fetched using the getAuthStatus(), getAuthValue(), getFileAuthCode(), isTransferredFile(), getStreamErrorCode(), and getStreamErrorMessage() methods.

1.11.1.2 downloadFile(FileTransferProgressListener) Method

Download the file specified by the properties of this object with progress events posted to the specified listener.

Syntax

```
public abstract boolean downloadFile (FileTransferProgressListener listener)
throws ULjException
```

Parameters

listener The object that receives file transfer progress events.

Returns

True if the download is successful; otherwise, a ULjException is thrown, and the method does not return normally.

Remarks

Errors may result in no data being sent to the listener.

Related Information

downloadFile() Method [page 81]

1.11.2 getAuthenticationParms() Method

Returns parameters provided to a custom user authentication script.

Syntax

```
public abstract String getAuthenticationParms ()
```

Returns

The list of authentication parms or null if no parameters are specified.

Related Information

setAuthenticationParms(String) Method [page 90]

1.11.3 getAuthStatus() Method

Returns the authorization status code of the last file transfer attempt.

Syntax

```
public abstract int getAuthStatus ()
```

Returns

An AuthStatusCode class value.

1.11.4 getAuthValue() Method

Returns the value specified in custom user authentication synchronization scripts.

Syntax

```
public abstract long getAuthValue ()
```

Returns

An integer returned from custom user authentication synchronization scripts.

1.11.5 getFileAuthCode() Method

Returns the return value from the authenticate_file_transfer script for the last file transfer attempt.

Syntax

```
public abstract int getFileAuthCode ()
```

Returns

An integer returned from the authenticate_file_transfer script for the last file transfer attempt.

1.11.6 getLivenessTimeout() Method

Returns the liveness timeout length, in seconds.

Syntax

```
public abstract int getLivenessTimeout ()
```

Returns

The timeout.

Related Information

setLivenessTimeout(int) Method [page 91]

1.11.7 getLocalFileName() Method

Determines the local file name.

Syntax

```
public abstract String getLocalFileName ()
```

Returns

The local file name for the downloaded file.

Remarks

For file downloads, this is the name of the downloaded file. For file uploads, this is the name of the file to upload.

Related Information

setLocalFileName(String) Method [page 92]

1.11.8 getLocalPath() Method

Specifies where to find or store the file in the local file system.

Syntax

```
public abstract String getLocalPath ()
```

Returns

The local directory.

Related Information

setLocalPath(String) Method [page 92]

1.11.9 getPassword() Method

Returns the MobiLink password for the user specified with the setUserName method.

Syntax

```
public abstract String getPassword ()
```

Returns

The password for the MobiLink user.

Related Information

setPassword(String) Method [page 93]

1.11.10 getRemoteKey() Method

Determines the current remote key value.

Syntax

```
public abstract String getRemoteKey ()
```

Returns

The remote key value or null if the remote key is unspecified.

Related Information

setRemoteKey(String) Method [page 94]

1.11.11 getServerFileName() Method

Returns the name of the file on the server.

Syntax

```
public abstract String getServerFileName ()
```

Returns

The name of the file in the server side.

Remarks

For file downloads, this is the name of the file to download. For file uploads, this is the name of the uploaded file.

Related Information

setServerFileName(String) Method [page 95]

1.11.12 getStreamErrorCode() Method

Returns the error code reported by the stream.

Syntax

```
public abstract int getStreamErrorCode ()
```

Returns

0 if there was no communication stream error; otherwise, returns the response code from the server.

Remarks

The error code is the HTTP response code.

1.11.13 getStreamErrorMessage() Method

Returns the error message reported by the stream itself.

Syntax

```
public abstract String getStreamErrorMessage ()
```

Returns

Null, if no message is available; otherwise, returns the response message.

Remarks

This is the HTTP response message.

1.11.14 getStreamParms() Method

Returns the parameters used to configure the synchronization stream.

Syntax

```
public abstract StreamHTTPParms getStreamParms ()
```

Returns

A StreamTCPIPParms object specifying the parameters for synchronization stream.

Remarks

The synchronization stream type is specified when the FileTransfer object is created.

1.11.15 getUserName() Method

Returns the MobiLink user name that uniquely identifies the client to the MobiLink server.

Syntax

```
public abstract String getUserName ()
```

Returns

The MobiLink user name.

Related Information

setUserName(String) Method [page 96]

1.11.16 getVersion() Method

Returns the synchronization script to use.

Syntax

```
public abstract String getVersion ()
```

Returns

The script version.

Related Information

setVersion(String) Method [page 97]

1.11.17 isResumePartialTransfer() Method

Determines whether to resume or discard a previous partial transfer.

Syntax

```
public abstract boolean isResumePartialTransfer ()
```

Returns

True if to resume the download; otherwise, returns false.

Related Information

setResumePartialTransfer(boolean) Method [page 95]

1.11.18 isTransferredFile() Method

Checks whether the file was actually downloaded during the last file transfer attempt.

Syntax

```
public abstract boolean isTransferredFile ()
```

Returns

True if the file transferred; otherwise, returns false.

Remarks

If the file is already up-to-date when the transferFile() method is invoked, this method returns true while the isTransferredFile() method returns false.

If an error occurs and the transferFile() method throws an exception, the isTransferredFile() method returns false.

1.11.19 setAuthenticationParms(String) Method

Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Syntax

```
public abstract void setAuthenticationParms (String authParms) throws
ULjException
```

Parameters

authParms A comma separated list of authentication parameters, or the null reference. See the class description of the SyncParms class for more information about comma separated lists.

Remarks

Only the first 255 strings are used and each string should be no longer than 128 characters (longer strings are truncated when sent to MobiLink).

Related Information

getAuthenticationParms() Method [page 82]

1.11.20 setLivenessTimeout(int) Method

Sets the liveness timeout length, in seconds.

Syntax

```
public abstract void setLivenessTimeout (int timeout) throws ULjException
```

Parameters

timeout The new liveness timeout value.

Remarks

The liveness timeout is the length of time the server allows a remote to be idle. If the remote does not communicate with the server for 1 second, the server assumes that the remote has lost the connection, and terminates the file transfer. The remote automatically sends periodic messages to the server to keep the connection alive.

If a negative value is set, an exception is thrown. The value may be changed by the MobiLink server without notice. This change occurs if the value is set too low or too high.

The default value is 240 seconds.

Related Information

getLivenessTimeout() Method [page 84]

1.11.21 setLocalFileName(String) Method

Specifies the local file name.

Syntax

```
public abstract void setLocalFileName (String localFileName)
```

Parameters

localFileName A string specifying the local file name for the downloaded file. If the value is a null reference, fileName is used. The default is a null reference.

Remarks

For file downloads, this is the name of the downloaded file. For file uploads, this is the name of the file to upload. The file name must not include any drive of path information.

Related Information

getLocalFileName() Method [page 84]
setLocalPath(String) Method [page 92]

1.11.22 setLocalPath(String) Method

Specifies where to find or store the file in the local file system.

Syntax

```
public abstract void setLocalPath (String localPath)
```

Parameters

localPath A string specifying the local directory of the file. The default is a null reference.

Remarks

The syntax of the local directory varies among platforms:

- For a desktop, the syntax is like "C:\\ulj\\"
- For an Android file system, the syntax is like "/sdcard/Android/data/your.package.name/files/"

The default local directory also varies depending on the device operating system:

- For a desktop, if the localPath parameter is null, the file is stored in the current directory
- For an Android file system store, the localPath parameter has no default value, and must be explicitly set.

Related Information

getLocalPath() Method [page 85]
setLocalFileName(String) Method [page 92]

1.11.23 setPassword(String) Method

Sets the MobiLink password for the user specified with the setUserName method.

Syntax

```
public abstract void setPassword (String password) throws ULjException
```

Parameters

password A password for the MobiLink user.

Remarks

This user name and password is separate from any database user ID and password. This method is used to authenticate the application against the MobiLink server.

The default is an empty string, suggesting no password.

Related Information

getPassword() Method [page 85]
setUserName(String) Method [page 96]

1.11.24 setRemoteKey(String) Method

Specifies the remote key.

Syntax

```
public abstract void setRemoteKey (String remoteKey)
```

Parameters

remoteKey The remote key value or null to leave it unspecified.

Remarks

The remote key is a parameter passed to the server authenticate_file_upload script.

The script can use this parameter to determine the name and location of the file to be stored on the server.

If this value is unspecified, the getUserName() value is used as the remote key.

Related Information

getRemoteKey() Method [page 86]

1.11.25 setResumePartialTransfer(boolean) Method

Specifies whether to resume or discard a previous partial transfer.

Syntax

```
public abstract void setResumePartialTransfer (boolean resume)
```

Parameters

resume Set to true to resume a previous partial download, or false to discard a previous partial download.

Remarks

The default is true.

UltraLiteJ has the ability to restart file transfers that fail because of communication errors or user aborts through the FileTransferProgressListener object.

For file downloads, UltraLiteJ processes the download as it is received. If a download is interrupted, then the partially download file is retained and can be resumed during the next file transfer. If the file has been updated on the server, the partial download is discarded and a new download started.

For file uploads, the MobiLink server keeps partially uploaded files so that a subsequent file upload can resume a previous one. However, if the file has been updated locally, the partial upload is discarded and a new upload is started.

Related Information

isResumePartialTransfer() Method [page 89]

1.11.26 setServerFileName(String) Method

Specifies the name of the file on the server.

Syntax

```
public abstract void setServerFileName (String fileName) throws ULjException
```

Parameters

fileName A string specifying the name of the file as recognized by the MobiLink server.

Remarks

For file downloads, this is the name of the file to download. For file uploads, this is the name of the uploaded file.

This parameter is initialized when the FileTransfer object is created.

MobiLink first searches for the file in the userName subdirectory followed by the root directory. The root download directory is specified via the MobiLink server's -ftr option, and the root upload directory is specified via the -ftru option.

fileName must not include any drive or path information, or the MobiLink server will be unable to find it. For example, "myfile.txt" is valid, but "somedir\myfile.txt", "..\myfile.txt", and "c:\myfile.txt" are all invalid.

Related Information

getServerFileName() Method [page 86]

1.11.27 setUserName(String) Method

Sets the MobiLink user name that uniquely identifies the client to the MobiLink server.

Syntax

```
public abstract void setUserName (String userName) throws ULjException
```

Parameters

userName The MobiLink user name.

Remarks

The MobiLink server uses this value to locate the file on the server side. The MobiLink user name and password are separate from any database user ID and password, and serve to identify and authenticate the application to the MobiLink server.

This parameter is initialized when the FileTransfer object is created.

Related Information

getUserName() Method [page 88]
setPassword(String) Method [page 93]

1.11.28 setVersion(String) Method

Sets the synchronization script to use.

Syntax

```
public abstract void setVersion (String version) throws ULjException
```

Parameters

version The script version.

Remarks

Each synchronization script in the consolidated database is marked with version string. The version string allows an UltraLiteJ application to choose from a set of synchronization scripts.

This parameter is initialized when the FileTransfer object is created.

Related Information

getVersion() Method [page 89]

1.11.29 uploadFile Method

Uploads the file with the specified properties of this object.

Overload list

Modifier and Type	Overload name	Description
public abstract boolean	uploadFile() [page 98]	Uploads the file with the specified properties of this object.
public abstract boolean	uploadFile(FileTransferProgressListener) [page 99]	Upload the file specified by the properties of this object with progress events posted to the specified listener.

In this section:

uploadFile() Method [page 98]
Uploads the file with the specified properties of this object.

uploadFile(FileTransferProgressListener) Method [page 99]
Upload the file specified by the properties of this object with progress events posted to the specified listener.

1.11.29.1 uploadFile() Method

Uploads the file with the specified properties of this object.

Syntax

```
public abstract boolean uploadFile () throws ULjException
```

Returns

True if the upload is successful; otherwise, a ULjException is thrown and the method does not return normally.

Remarks

The file specified by the setLocalFileName and setLocalPath methods is uploaded to the MobiLink server to the file specified by the setServerFileName method using the specified stream, userName, password, and script version.

Further options can be specified using the setAuthenticationParms() and setResumePartialTransfer() methods.

A detailed result status can be fetched using the getAuthStatus(), getAuthValue(), getFileAuthCode(), isTransferredFile(), getStreamErrorCode(), and getStreamErrorMessage() methods.

1.11.29.2 uploadFile(FileTransferProgressListener) Method

Upload the file specified by the properties of this object with progress events posted to the specified listener.

Syntax

```
public abstract boolean uploadFile (FileTransferProgressListener listener)
throws ULjException
```

Parameters

listener The object that receives file transfer progress events.

Returns

True if the upload is successful; otherwise, a ULjException is thrown and the method does not return normally.

Remarks

Errors may result in no data being sent to the listener.

Related Information

uploadFile() Method [page 98]

1.12 FileTransferProgressData Interface

Reports file transfer progress monitoring data.

Syntax

```
public interface FileTransferProgressData
```

Members

All members of FileTransferProgressData, including inherited members.

Methods

Modifier and Type	Method	Description
public abstract long	getBytesTransferred() [page 100]	Returns the number of bytes transferred so far.
public abstract long	getFileSize() [page 101]	Returns the size of the file being transferred.
public abstract long	getResumedAtSize() [page 102]	Returns the point in the file where the transfer was resumed.

In this section:

getBytesTransferred() Method [page 100]
: Returns the number of bytes transferred so far.

getFileSize() Method [page 101]
: Returns the size of the file being transferred.

getResumedAtSize() Method [page 102]
: Returns the point in the file where the transfer was resumed.

1.12.1 getBytesTransferred() Method

Returns the number of bytes transferred so far.

Syntax

```
public abstract long getBytesTransferred ()
```

Returns

The number of bytes transferred so far.

Remarks

This method counts the number of bytes transferred by the current file transfer session, and adds the bytes transferred by any previous interrupted transfers.

Subtract the value returned by the getResumedAtSize method to determine the number of bytes transferred by the current session.

Related Information

getResumedAtSize() Method [page 102]

1.12.2 getFileSize() Method

Returns the size of the file being transferred.

Syntax

```
public abstract long getFileSize ()
```

Returns

The size of the file in bytes.

Remarks

The returned value remains constant for the duration of the file transfer session.

1.12.3 getResumedAtSize() Method

Returns the point in the file where the transfer was resumed.

Syntax

```
public abstract long getResumedAtSize ()
```

Returns

The number of bytes previously transferred.

Remarks

The returned value remains constant for the duration of the file transfer session.

1.13 FileTransferProgressListener Interface

Receives file transfer progress events.

Syntax

```
public interface FileTransferProgressListener
```

Members

All members of FileTransferProgressListener, including inherited members.

Methods

Modifier and Type	Method	Description
public boolean	fileTransferProgressed(FileTransfer-ProgressData) [page 103]	Is invoked during a file transfer to in-form the user of the transfer progress.

Remarks

Create a new class to receive progress reports during a file transfer.

The following example illustrates a simple SyncObserver interface that implements the FileTransferProgressListener interface:

```
class MyObserver implements FileTransferProgressListener {
  public boolean fileTransferProgressed( FileTransferProgressData data ) {
      System.out.println(
          "file transfer progress "
          + " bytes received = " + data.getBytesTransferred()
      );
      return false;   // Always continue file transfer.
  }
  public MyObserver() {} // The default constructor.
}
```

In this section:

1.13.1 fileTransferProgressed(FileTransferProgressData) Method

Is invoked during a file transfer to inform the user of the transfer progress.

Syntax

```
public boolean fileTransferProgressed (FileTransferProgressData data)
```

Parameters

data A FileTransferProgressData object containing the latest file transfer progress data.

Returns

This method should return true to cancel the transfer; otherwise, return false to continue.

Remarks

The listener is called under the following conditions:

- Before the first disk write
- After every disk write or every 0.5 seconds, whichever is later
- After the file download is complete

Usually, the cancel request is accepted by the UltraLiteJ API. This results in a ULjException object being thrown with the errorCode set to the ULjException.SQLE_INTERRUPTED constant.

UltraLiteJ API methods should not be invoked during a fileTransferProgressed call.

1.14 IndexSchema Interface

Specifies the schema of an index and provides constants that are useful for querying system tables.

Syntax

```
public interface IndexSchema
```

Members

All members of IndexSchema, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final byte	ASCENDING	Denotes that an index is sorted in ascending order for a column.
public static final byte	DESCENDING	Denotes that an index is sorted in descending order for a column.
public static final byte	UNIQUE_KEY	Denotes that an index is a unique key. This value can be logically combined with other flags in the index_flags column of the SYS_INDEXES system table.
public static final byte	UNIQUE_INDEX	Denotes that an index is a unique index. This value can be logically combined with other flags in the index_flags column of the SYS_INDEXES system table.

Modifier and Type	Variable	Description
public static final byte	PERSISTENT	Denotes that an index is persistent. This value can be logically combined with other flags in the index_flags column of the SYS_INDEXES system table.
public static final byte	PRIMARY_INDEX	Denotes that an index is a primary key. This value can be logically combined with other flags in the index_flags column of the SYS_INDEXES system table.

Remarks

This interface only contains index-related constants, including flags and the sort order of indexes.

1.15 PreparedStatement Interface

Provides methods to execute a SQL query to generate a ResultSet object or to execute a prepared SQL statement on a database.

Syntax

```
public interface PreparedStatement
```

Members

All members of PreparedStatement, including inherited members.

Methods

Modifier and Type	Method	Description
public void	close() [page 108]	Closes the PreparedStatement to release the memory resources associated with it.
public boolean	execute() [page 108]	Executes the prepared SQL statement.
public ResultSet	executeQuery() [page 108]	Executes the prepared SQL SELECT statement and returns a ResultSet object.

Modifier and Type	Method	Description
public java.io.OutputStream	getBlobOutputStream [page 109]	Returns an OutputStream object.
public java.io.Writer	getClobWriter [page 110]	Returns a Writer object.
public int	getOrdinal(String) [page 112]	Returns the (base-one) ordinal for the value represented by the name.
public short	getParameterCount() [page 112]	Gets the number of input parameters for this statement.
public short	getParameterType(int) [page 112]	Gets the domain type of a parameter.
public String	getPlan() [page 113]	Returns a text-based description of the SQL query execution plan.
public String	getPlanTree() [page 114]	Returns a text-based description of the SQL query execution plan, represented as a tree.
public ResultSet	getResultSet() [page 115]	Returns the ResultSet object for a prepared SQL statement.
public int	getUpdateCount() [page 115]	Returns the number of rows inserted, updated or deleted since the last execute statement.
public boolean	hasResultSet() [page 116]	Determines if the PreparedStatement object contains a ResultSet object.
public void	set [page 116]	Sets a value to the host variable in the SQL statement.
public void	setNull [page 128]	Sets a null value to the host variable in the SQL statement that is defined by name.

Remarks

The following example demonstrates how to create a PreparedStatement object, check if a SELECT statement execution creates a ResultSet object, save any ResultSet object to a local variable, and then close the PreparedStatement:

```
// Create a new PreparedStatement object from an existing connection.
String sql_string = "SELECT * FROM SampleTable";
PreparedStatement ps = conn.prepareStatement(sql_string);
// Result returns true if the statement runs successfully.
boolean result = ps.execute();
// Check if the PreparedStatement object contains a ResultSet object.
if (ps.hasResultSet()) {
    // Store the ResultSet in the rs variable.
    ResultSet rs = ps.getResultSet;
}
// Close the PreparedStatement object to release resources.
ps.close();
```

When a statement contains expressions, it may contain a host variable wherever a column name could appear. Host variables are entered as either a *?* character (unnamed host variables) or a *:name* (named host variable).

In the following example, there are two host variables that may be set using the PreparedStatement object, and were prepared for the SQL statement in question:

```
SELECT * FROM SampleTable WHERE pk > :bound AND pk < ?
```

In this section:

Related Information

Connection Interface [page 31]
prepareStatement(String) Method [page 50]

1.15.1 close() Method

Closes the PreparedStatement to release the memory resources associated with it.

Syntax

```
public void close () throws ULjException
```

Remarks

No further methods can be used on this object. If the PreparedStatement object contains a ResultSet object, both objects are closed.

1.15.2 execute() Method

Executes the prepared SQL statement.

Syntax

```
public boolean execute () throws ULjException
```

Returns

True if the execute statement runs successfully; otherwise, returns false.

1.15.3 executeQuery() Method

Executes the prepared SQL SELECT statement and returns a ResultSet object.

Syntax

```
public ResultSet executeQuery () throws ULjException
```

Returns

The ResultSet object containing the query result of the prepared SQL SELECT statement.

Related Information

ResultSet Interface [page 129]

1.15.4 getBlobOutputStream Method

Returns an OutputStream object.

Overload list

Modifier and Type	Overload name	Description
public java.io.OutputStream	getBlobOutputStream(String) [page 109]	Returns an OutputStream object.
public java.io.OutputStream	getBlobOutputStream(int) [page 110]	Returns an OutputStream object.

In this section:

1.15.4.1 getBlobOutputStream(String) Method

Returns an OutputStream object.

Syntax

```
public java.io.OutputStream getBlobOutputStream (String name) throws
ULjException
```

Parameters

name A String representing the host variable name.

Returns

The OutputStream object for the named value.

1.15.4.2 getBlobOutputStream(int) Method

Returns an OutputStream object.

Syntax

```
public java.io.OutputStream getBlobOutputStream (int ordinal) throws
ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

Returns

The OutputStream object for the named value.

1.15.5 getClobWriter Method

Returns a Writer object.

Overload list

Modifier and Type	Overload name	Description
public java.io.Writer	getClobWriter(String) [page 111]	Returns a Writer object.
public java.io.Writer	getClobWriter(int) [page 111]	Returns a Writer object.

In this section:

getClobWriter(String) Method [page 111]
Returns a Writer object.

getClobWriter(int) Method [page 111]
  Returns a Writer object.

1.15.5.1 getClobWriter(String) Method

Returns a Writer object.

Syntax

```
public java.io.Writer getClobWriter (String name) throws ULjException
```

Parameters

name A String representing the host variable name.

Returns

The Writer object for the named value.

1.15.5.2 getClobWriter(int) Method

Returns a Writer object.

Syntax

```
public java.io.Writer getClobWriter (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

Returns

The Writer object for the named value.

1.15.6 getOrdinal(String) Method

Returns the (base-one) ordinal for the value represented by the name.

Syntax

```
public int getOrdinal (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The (base-one) ordinal for the value represented by the name.

1.15.7 getParameterCount() Method

Gets the number of input parameters for the prepared statement.

Syntax

```
public short getParameterCount () throws ULjException
```

Returns

The number of input parameters for the prepared statement.

1.15.8 getParameterType(int) Method

Gets the domain type of a parameter.

Syntax

```
public short getParameterType (int ordinal) throws ULjException
```

Parameters

ordinal The 1-based ordinal of the parameter.

Returns

The domain type of the specified parameter.

1.15.9 getPlan() Method

Returns a text-based description of the SQL query execution plan.

Syntax

```
public String getPlan () throws ULjException
```

Returns

The String representation of the plan.

Remarks

This method is intended for use during development.

This plan contains the same information as is presented by the getPlanTree method. The difference is in the presentation.

An empty string is returned if there is no plan. Plans exist when the prepared statement is a SQL query.

The plan shows the operations used to execute the query when the plan is obtained before the associated query has been executed. Additionally, the plan shows the number of rows that each operation produced when the plan is obtained after the query has been executed. This plan can be used to gain insight about the execution of the query.

The following is an example of a plan tree, expressed as a String. It is displayed on multiple lines with '|' characters to represent the structure.

```
SELECT * FROM tab1, tab2 WHERE col1 > pk2
row: 2 20 10 banana
row: 3 30 10 banana
row: 4 40 10 banana
row: 4 40 30 peach
```

```
row: 5 50 10 banana
row: 5 50 30 peach
row: 5 50 40 apple
plan: root:7(inner-join:7(table-scan:5[tab1,prime_key],index-scan:
7[tab2,prime_key]))
```

Related Information

getPlanTree() Method [page 114]

1.15.10 getPlanTree() Method

Returns a text-based description of the SQL query execution plan, represented as a tree.

Syntax

```
public String getPlanTree () throws ULjException
```

Returns

The String representation of the plan, represented as a tree.

Remarks

This method is intended for use during development.

This plan contains the same information as is presented by the getPlan method. The difference is in the presentation.

An empty string is returned if there is no plan. Plans exist when the prepared statement is a SQL query.

The plan shows the operations used to execute the query when the plan is obtained before the associated query has been executed. Additionally, the plan shows the number of rows that each operation produced when the plan is obtained after the query has been executed. This plan can be used to gain insight about the execution of the query.

The following is an example of a plan tree, expressed as a String. It is displayed on multiple lines with '|' characters to represent the structure.

```
SELECT * FROM tab1, tab2 WHERE col1 > pk2
row: 2 20 10 banana
row: 3 30 10 banana
row: 4 40 10 banana
row: 4 40 30 peach
```

```
row: 5 50 10 banana
row: 5 50 30 peach
row: 5 50 40 apple
plan:
root:7
|
inner-join:7
| |
| index-scan:7[tab2,prime_key]
|
table-scan:5[tab1,prime_key]
```

Related Information

getPlan() Method [page 113]

1.15.11 getResultSet() Method

Returns the ResultSet object for a prepared SQL statement.

Syntax

```
public ResultSet getResultSet () throws ULjException
```

Returns

The ResultSet object containing the query result of the prepared SQL statement.

Related Information

ResultSet Interface [page 129]

1.15.12 getUpdateCount() Method

Returns the number of rows inserted, updated or deleted since the last execute statement.

Syntax

```
public int getUpdateCount () throws ULjException
```

Returns

-1 when a statement cannot perform changes; otherwise, returns the number of rows that have been changed.

1.15.13 hasResultSet() Method

Determines if the PreparedStatement object contains a ResultSet object.

Syntax

```
public boolean hasResultSet () throws ULjException
```

Returns

True if a ResultSet was found; otherwise, returns false.

Related Information

ResultSet Interface [page 129]

1.15.14 set Method

Sets a value to the host variable in the SQL statement.

Overload list

Modifier and Type	Overload name	Description
public void	set(String, Date) [page 119]	Sets a java.util.Date to the host variable in the SQL statement that is defined by name.
public void	set(String, DecimalNumber) [page 119]	Sets a DecimalNumber object to the host variable in the SQL statement that is defined by name.

Modifier and Type	Overload name	Description
public void	set(String, String) [page 120]	Sets a String value to the host variable in the SQL statement that is defined by name.
public void	set(String, UUIDValue) [page 120]	Sets a UUIDValue value to the host vari-able in the SQL statement that is de-fined by name.
public void	set(String, boolean) [page 121]	Sets a boolean value to the host varia-ble in the SQL statement that is defined by name.
public void	set(String, byte[]) [page 121]	Sets a byte array value to the host vari-able in the SQL statement that is de-fined by name.
public void	set(String, double) [page 121]	Sets a double value to the host variable in the SQL statement that is defined by name.
public void	set(String, float) [page 122]	Sets a float value to the host variable in the SQL statement that is defined by name.
public void	set(String, int) [page 122]	Sets an integer value to the host varia-ble in the SQL statement that is defined by name.
public void	set(String, long) [page 123]	Sets a long integer value to the host variable in the SQL statement that is defined by name.
public void	set(int, Date) [page 123]	Sets a java.util.Date to the host variable in the SQL statement that is defined by ordinal.
public void	set(int, DecimalNumber) [page 124]	Sets a DecimalNumber object to the host variable in the SQL statement that is defined by ordinal.
public void	set(int, String) [page 124]	Sets a String value to the host variable in the SQL statement that is defined by ordinal.
public void	set(int, UUIDValue) [page 124]	Sets a UUIDValue value to the host vari-able in the SQL statement that is de-fined by ordinal.
public void	set(int, boolean) [page 125]	Sets a boolean value to the host varia-ble in the SQL statement that is defined by ordinal.
public void	set(int, byte[]) [page 125]	Sets a byte array value to the host vari-able in the SQL statement that is de-fined by ordinal.
public void	set(int, double) [page 126]	Sets a double value to the host variable in the SQL statement that is defined by ordinal.

Modifier and Type	Overload name	Description
public void	set(int, float) [page 126]	Sets a float value to the host variable in the SQL statement that is defined by ordinal.
public void	set(int, int) [page 127]	Sets an integer value to the host variable in the SQL statement that is defined by ordinal.
public void	set(int, long) [page 127]	Sets a long integer value to the host variable in the SQL statement that is defined by ordinal.

In this section:

set(String, Date) Method [page 119]
Sets a java.util.Date to the host variable in the SQL statement that is defined by name.

set(String, DecimalNumber) Method [page 119]
Sets a DecimalNumber object to the host variable in the SQL statement that is defined by name.

set(String, String) Method [page 120]
Sets a String value to the host variable in the SQL statement that is defined by name.

set(String, UUIDValue) Method [page 120]
Sets a UUIDValue value to the host variable in the SQL statement that is defined by name.

set(String, boolean) Method [page 121]
Sets a boolean value to the host variable in the SQL statement that is defined by name.

set(String, byte[]) Method [page 121]
Sets a byte array value to the host variable in the SQL statement that is defined by name.

set(String, double) Method [page 121]
Sets a double value to the host variable in the SQL statement that is defined by name.

set(String, float) Method [page 122]
Sets a float value to the host variable in the SQL statement that is defined by name.

set(String, int) Method [page 122]
Sets an integer value to the host variable in the SQL statement that is defined by name.

set(String, long) Method [page 123]
Sets a long integer value to the host variable in the SQL statement that is defined by name.

set(int, Date) Method [page 123]
Sets a java.util.Date to the host variable in the SQL statement that is defined by ordinal.

set(int, DecimalNumber) Method [page 124]
Sets a DecimalNumber object to the host variable in the SQL statement that is defined by ordinal.

set(int, String) Method [page 124]
Sets a String value to the host variable in the SQL statement that is defined by ordinal.

set(int, UUIDValue) Method [page 124]
Sets a UUIDValue value to the host variable in the SQL statement that is defined by ordinal.

set(int, boolean) Method [page 125]
Sets a boolean value to the host variable in the SQL statement that is defined by ordinal.

set(int, byte[]) Method [page 125]
: Sets a byte array value to the host variable in the SQL statement that is defined by ordinal.

set(int, double) Method [page 126]
: Sets a double value to the host variable in the SQL statement that is defined by ordinal.

set(int, float) Method [page 126]
: Sets a float value to the host variable in the SQL statement that is defined by ordinal.

set(int, int) Method [page 127]
: Sets an integer value to the host variable in the SQL statement that is defined by ordinal.

set(int, long) Method [page 127]
: Sets a long integer value to the host variable in the SQL statement that is defined by ordinal.

1.15.14.1 set(String, Date) Method

Sets a java.util.Date to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    java.util.Date value
) throws ULjException
```

Parameters

name A String representing the host variable name.
value The value to be set.

1.15.14.2 set(String, DecimalNumber) Method

Sets a DecimalNumber object to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    DecimalNumber value
) throws ULjException
```

Parameters

name A String representing the host variable name.
value The DecimalNumber value to be set.

1.15.14.3 set(String, String) Method

Sets a String value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    String value
) throws ULjException
```

Parameters

name A String representing the host variable name.
value The value to be set.

1.15.14.4 set(String, UUIDValue) Method

Sets a UUIDValue value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    UUIDValue value
) throws ULjException
```

Parameters

name A String representing the host variable name.
value The value to be set.

1.15.14.5 set(String, boolean) Method

Sets a boolean value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    boolean value
) throws ULjException
```

Parameters

name A String representing the host variable name.

value The value to be set.

1.15.14.6 set(String, byte[]) Method

Sets a byte array value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    byte[] value
) throws ULjException
```

Parameters

name A String representing the host variable name.

value The value to be set.

1.15.14.7 set(String, double) Method

Sets a double value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    double value
) throws ULjException
```

Parameters

name A String representing the host variable name.

value The value to be set.

1.15.14.8 set(String, float) Method

Sets a float value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    float value
) throws ULjException
```

Parameters

name A String representing the host variable name.

value The value to be set.

1.15.14.9 set(String, int) Method

Sets an integer value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    int value
) throws ULjException
```

Parameters

name A String representing the host variable name.

value The value to be set.

1.15.14.10 set(String, long) Method

Sets a long integer value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void set (
    String name,
    long value
) throws ULjException
```

Parameters

name A String representing the host variable name.

value The value to be set.

1.15.14.11 set(int, Date) Method

Sets a java.util.Date to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    java.util.Date value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

value The value to be set.

1.15.14.12 set(int, DecimalNumber) Method

Sets a DecimalNumber object to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    DecimalNumber value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.
value The DecimalNumber value to be set.

1.15.14.13 set(int, String) Method

Sets a String value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    String value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.
value The value to be set.

1.15.14.14 set(int, UUIDValue) Method

Sets a UUIDValue value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    UUIDValue value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

value The value to be set.

1.15.14.15 set(int, boolean) Method

Sets a boolean value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    boolean value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

value The value to be set.

1.15.14.16 set(int, byte[]) Method

Sets a byte array value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    byte[] value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

value The value to be set.

1.15.14.17 set(int, double) Method

Sets a double value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    double value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.
value The value to be set.

1.15.14.18 set(int, float) Method

Sets a float value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    float value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.
value The value to be set.

1.15.14.19 set(int, int) Method

Sets an integer value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    int value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.
value The value to be set.

1.15.14.20 set(int, long) Method

Sets a long integer value to the host variable in the SQL statement that is defined by ordinal.

Syntax

```
public void set (
    int ordinal,
    long value
) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.
value The value to be set.

1.15.15 setNull Method

Sets a null value to the host variable in the SQL statement that is defined by name.

Overload list

Modifier and Type	Overload name	Description
public void	setNull(String) [page 128]	Sets a null value to the host variable in the SQL statement that is defined by name.
public void	setNull(int) [page 129]	Sets a null value to the host variable in the SQL statement.

In this section:

setNull(String) Method [page 128]
    Sets a null value to the host variable in the SQL statement that is defined by name.

setNull(int) Method [page 129]
    Sets a null value to the host variable in the SQL statement.

1.15.15.1 setNull(String) Method

Sets a null value to the host variable in the SQL statement that is defined by name.

Syntax

```
public void setNull (String name) throws ULjException
```

Parameters

name A String representing the host variable name.

1.15.15.2 setNull(int) Method

Sets a null value to the host variable in the SQL statement.

Syntax

```
public void setNull (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the host variable as ordered in the SQL statement.

1.16 ResultSet Interface

Provides methods to traverse a table by row, and access the column data.

Syntax

```
public interface ResultSet
```

Members

All members of ResultSet, including inherited members.

Methods

Modifier and Type	Method	Description
public boolean	afterLast() [page 132]	Moves the cursor after the last row.
public boolean	beforeFirst() [page 133]	Moves the cursor before the first row.
public void	close() [page 133]	Closes the ResultSet object to release the memory resources associated with it.
public boolean	first() [page 133]	Moves the cursor to the first row.
public java.io.InputStream	getBlobInputStream [page 134]	Returns an InputStream object for long binary types, including file-based long binaries.
public boolean	getBoolean [page 135]	Returns a boolean value.
public byte[]	getBytes [page 137]	Returns a byte array.

Modifier and Type	Method	Description
public java.io.Reader	getClobReader [page 138]	Returns a Reader object.
public java.util.Date	getDate [page 140]	Returns a java.util.Date object.
public DecimalNumber	getDecimalNumber [page 141]	Returns a DecimalNumber object.
public double	getDouble [page 143]	Returns a double value based on the column name.
public float	getFloat [page 144]	Returns a float value.
public int	getInt [page 146]	Returns an integer value.
public long	getLong [page 147]	Returns a long integer value.
public int	getOrdinal(String) [page 149]	Returns the (base-one) ordinal for the value represented by a String.
public ResultSetMetadata	getResultSetMetadata() [page 149]	Returns a ResultSetMetadata object that contains the metadata for the Re-sultSet object.
public long	getRowCount(long) [page 150]	Gets the number of rows in the table.
public int	getSize [page 151]	Gets the actual size of a result set col-umn.
public String	getString [page 152]	Returns a String value.
public UUIDValue	getUUIDValue [page 154]	Returns a UUIDValue object.
public boolean	isNull [page 155]	Tests if the value at the specified col-umn name is null.
public boolean	last() [page 157]	Moves the cursor to the last row.
public boolean	next() [page 157]	Fetches the next row of data in the Re-sultSet object.
public boolean	previous() [page 158]	Fetches the previous row of data in the ResultSet object.
public boolean	relative(int) [page 158]	Moves the cursor by an offset of rows from the current cursor position.

Remarks

A ResultSet object is generated when the execute or executeQuery method is called on a PreparedStatement object with a SQL SELECT statement.

The following example demonstrates how to fetch a row in the ResultSet object, and access data from a specified column:

```
// Define a new SQL SELECT statement.
String sql_string = "SELECT column1, column2 FROM SampleTable";
// Create a new PreparedStatement from an existing connection.
PreparedStatement ps = conn.prepareStatement(sql_string);
// Create a new ResultSet to contain the query results of the SQL statement.
ResultSet rs = ps.executeQuery();
// Check if the PreparedStatement contains a ResultSet.
```

```
if (ps.hasResultSet()) {
    // Retrieve the column1 value from the first row using getString.
    String row1_col1 = rs.getString(1);
    // Get the next row in the table.
    if (rs.next) {
         // Retrieve the value of column1 from the second row.
         String row2_col1 = rs.getString(1);
    }
}
rs.close();
ps.close();
```

In this section:

getRowCount(long) Method [page 150]
: Gets the number of rows in the table.

getSize Method [page 151]
: Gets the actual size of a result set column.

getString Method [page 152]
: Returns a String value.

getUUIDValue Method [page 154]
: Returns a UUIDValue object.

isNull Method [page 155]
: Tests if the value at the specified column name is null.

last() Method [page 157]
: Moves the cursor to the last row.

next() Method [page 157]
: Fetches the next row of data in the ResultSet object.

previous() Method [page 158]
: Fetches the previous row of data in the ResultSet object.

relative(int) Method [page 158]
: Moves the cursor by an offset of rows from the current cursor position.

Related Information

PreparedStatement Interface [page 105]
execute() Method [page 108]
executeQuery() Method [page 108]
Connection Interface [page 31]

1.16.1 afterLast() Method

Moves the cursor after the last row.

Syntax

```
public boolean afterLast () throws ULjException
```

Returns

True on success; otherwise, returns false.

1.16.2 beforeFirst() Method

Moves the cursor before the first row.

Syntax

```
public boolean beforeFirst () throws ULjException
```

Returns

True on success; otherwise, returns false.

1.16.3 close() Method

Closes the ResultSet object to release the memory resources associated with it.

Syntax

```
public void close () throws ULjException
```

Remarks

Subsequent attempts to fetch rows from a closed ResultSet object throw an error.

1.16.4 first() Method

Moves the cursor to the first row.

Syntax

```
public boolean first () throws ULjException
```

Returns

True on success; otherwise, returns false.

1.16.5 getBlobInputStream Method

Returns an InputStream object for long binary types, including file-based long binaries.

Overload list

Modifier and Type	Overload name	Description
public java.io.InputStream	getBlobInputStream(String) [page 134]	Returns an InputStream object for long binary types, including file-based long binaries.
public java.io.InputStream	getBlobInputStream(int) [page 135]	Returns an InputStream object for long binary types, including file-based long binaries.

In this section:

1.16.5.1 getBlobInputStream(String) Method

Returns an InputStream object for long binary types, including file-based long binaries.

Syntax

```
public java.io.InputStream getBlobInputStream (String name) throws
ULjException
```

Parameters

name A String representing the table column name.

Returns

The InputStream object representation of the named value.

1.16.5.2 getBlobInputStream(int) Method

Returns an InputStream object for long binary types, including file-based long binaries.

Syntax

```
public java.io.InputStream getBlobInputStream (int ordinal) throws
ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The InputStream object representation of the named value.

1.16.6 getBoolean Method

Returns a boolean value.

Overload list

Modifier and Type	Overload name	Description
public boolean	getBoolean(String) [page 136]	Returns a boolean value.
public boolean	getBoolean(int) [page 136]	Returns a boolean value.

In this section:

getBoolean(String) Method [page 136]
Returns a boolean value.

getBoolean(int) Method [page 136]
Returns a boolean value.

1.16.6.1 getBoolean(String) Method

Returns a boolean value.

Syntax

```
public boolean getBoolean (String name) throws ULjException
```

Parameters

name A String representing the table column name in the ResultSet object.

Returns

The boolean representation of the named value.

1.16.6.2 getBoolean(int) Method

Returns a boolean value.

Syntax

```
public boolean getBoolean (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The boolean representation of the named value.

1.16.7 getBytes Method

Returns a byte array.

Overload list

Modifier and Type	Overload name	Description
public byte[]	getBytes(String) [page 137]	Returns a byte array.
public byte[]	getBytes(int) [page 138]	Returns a byte array.

In this section:

getBytes(String) Method [page 137]
Returns a byte array.

getBytes(int) Method [page 138]
Returns a byte array.

1.16.7.1 getBytes(String) Method

Returns a byte array.

Syntax

```
public byte[] getBytes (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The byte array representation of the named value.

1.16.7.2 getBytes(int) Method

Returns a byte array.

Syntax

```
public byte[] getBytes (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The byte array representation of the named value.

1.16.8 getClobReader Method

Returns a Reader object.

Overload list

Modifier and Type	Overload name	Description
public java.io.Reader	getClobReader(String) [page 139]	Returns a Reader object.
public java.io.Reader	getClobReader(int) [page 139]	Returns a Reader object.

In this section:

getClobReader(String) Method [page 139]
Returns a Reader object.

getClobReader(int) Method [page 139]
Returns a Reader object.

1.16.8.1 getClobReader(String) Method

Returns a Reader object.

Syntax

```
public java.io.Reader getClobReader (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The Reader object representation of the named value.

1.16.8.2 getClobReader(int) Method

Returns a Reader object.

Syntax

```
public java.io.Reader getClobReader (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The Reader object representation of the named value.

1.16.9 getDate Method

Returns a java.util.Date object.

Overload list

Modifier and Type	Overload name	Description
public java.util.Date	getDate(String) [page 140]	Returns a java.util.Date object.
public java.util.Date	getDate(int) [page 141]	Returns a java.util.Date object.

In this section:

1.16.9.1 getDate(String) Method

Returns a java.util.Date object.

Syntax

```
public java.util.Date getDate (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The java.util.date object representation of the named value.

1.16.9.2 getDate(int) Method

Returns a java.util.Date object.

Syntax

```
public java.util.Date getDate (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The java.util.Date object representation of the named value.

1.16.10 getDecimalNumber Method

Returns a DecimalNumber object.

Overload list

Modifier and Type	Overload name	Description
public DecimalNumber	getDecimalNumber(String) [page 142]	Returns a DecimalNumber object.
public DecimalNumber	getDecimalNumber(int) [page 142]	Returns a DecimalNumber object.

In this section:

getDecimalNumber(String) Method [page 142]
Returns a DecimalNumber object.

getDecimalNumber(int) Method [page 142]
Returns a DecimalNumber object.

1.16.10.1 getDecimalNumber(String) Method

Returns a DecimalNumber object.

Syntax

```
public DecimalNumber getDecimalNumber (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The DecimalNumber object representation of the named value.

Related Information

DecimalNumber Interface [page 67]

1.16.10.2 getDecimalNumber(int) Method

Returns a DecimalNumber object.

Syntax

```
public DecimalNumber getDecimalNumber (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The DecimalNumber object representation of the named value.

Related Information

DecimalNumber Interface [page 67]

1.16.11 getDouble Method

Returns a double value based on the column name.

Overload list

Modifier and Type	Overload name	Description
public double	getDouble(String) [page 143]	Returns a double value based on the column name.
public double	getDouble(int) [page 144]	Returns a double value based on the column number.

In this section:

1.16.11.1 getDouble(String) Method

Returns a double value based on the column name.

Syntax

```
public double getDouble (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The double representation of the named value.

1.16.11.2 getDouble(int) Method

Returns a double value based on the column number.

Syntax

```
public double getDouble (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The double representation of the named value.

1.16.12 getFloat Method

Returns a float value.

Overload list

Modifier and Type	Overload name	Description
public float	getFloat(String) [page 145]	Returns a float value.

Modifier and Type	Overload name	Description
public float	getFloat(int) [page 145]	Returns a float value.

In this section:

1.16.12.1 getFloat(String) Method

Returns a float value.

Syntax

```
public float getFloat (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The float representation of the named value.

1.16.12.2 getFloat(int) Method

Returns a float value.

Syntax

```
public float getFloat (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The float representation of the named value.

1.16.13 getInt Method

Returns an integer value.

Overload list

Modifier and Type	Overload name	Description
public int	getInt(String) [page 146]	Returns an integer value.
public int	getInt(int) [page 147]	Returns an integer value.

In this section:

1.16.13.1 getInt(String) Method

Returns an integer value.

Syntax

```
public int getInt (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The integer representation of the named value.

1.16.13.2 getInt(int) Method

Returns an integer value.

Syntax

```
public int getInt (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The integer representation of the named value.

1.16.14 getLong Method

Returns a long integer value.

Overload list

Modifier and Type	Overload name	Description
public long	getLong(String) [page 148]	Returns a long integer value.

Modifier and Type	Overload name	Description
public long	getLong(int) [page 148]	Returns a long integer value.

In this section:

1.16.14.1 getLong(String) Method

Returns a long integer value.

Syntax

```
public long getLong (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The long integer representation of the named value.

1.16.14.2 getLong(int) Method

Returns a long integer value.

Syntax

```
public long getLong (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The long integer representation of the named value.

1.16.15 getOrdinal(String) Method

Returns the (base-one) ordinal for the value represented by a String.

Syntax

```
public int getOrdinal (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The ordinal value.

1.16.16 getResultSetMetadata() Method

Returns a ResultSetMetadata object that contains the metadata for the ResultSet object.

Syntax

```
public ResultSetMetadata getResultSetMetadata () throws ULjException
```

Returns

The ResultSetMetadata object.

1.16.17 getRowCount(long) Method

Gets the number of rows in the table.

Syntax

```
public long getRowCount (long threshold) throws ULjException
```

Parameters

threshold The limit on the number of rows to count. 0 indicates no limit.

Returns

The number of rows in the table.

Remarks

This method is equivalent to executing "SELECT COUNT(*) FROM table".

1.16.18 getSize Method

Gets the actual size of a result set column.

Overload list

Modifier and Type	Overload name	Description
public int	getSize(String) [page 151]	Gets the actual size of a result set col-umn.
public int	getSize(int) [page 152]	Gets the actual size of a result set col-umn.

In this section:

getSize(String) Method [page 151]
Gets the actual size of a result set column.

getSize(int) Method [page 152]
Gets the actual size of a result set column.

1.16.18.1 getSize(String) Method

Gets the actual size of a result set column.

Syntax

```
public int getSize (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The actual size of a result set column.

1.16.18.2 getSize(int) Method

Gets the actual size of a result set column.

Syntax

```
public int getSize (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The actual size of a result set column.

1.16.19 getString Method

Returns a String value.

Overload list

Modifier and Type	Overload name	Description
public String	getString(String) [page 153]	Returns a String value.
public String	getString(int) [page 153]	Returns a String value.

In this section:

getString(String) Method [page 153]
Returns a String value.

getString(int) Method [page 153]
Returns a String value.

1.16.19.1 getString(String) Method

Returns a String value.

Syntax

```
public String getString (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The String representation of the named value.

1.16.19.2 getString(int) Method

Returns a String value.

Syntax

```
public String getString (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The String representation of the named value.

1.16.20 getUUIDValue Method

Returns a UUIDValue object.

Overload list

Modifier and Type	Overload name	Description
public UUIDValue	getUUIDValue(String) [page 154]	Returns a UUIDValue object.
public UUIDValue	getUUIDValue(int) [page 155]	Returns a UUIDValue object.

In this section:

getUUIDValue(String) Method [page 154]
Returns a UUIDValue object.

getUUIDValue(int) Method [page 155]
Returns a UUIDValue object.

1.16.20.1 getUUIDValue(String) Method

Returns a UUIDValue object.

Syntax

```
public UUIDValue getUUIDValue (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

The UUIDValue object representation of the named value.

1.16.20.2 getUUIDValue(int) Method

Returns a UUIDValue object.

Syntax

```
public UUIDValue getUUIDValue (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

The UUIDValue object representation of the named value.

Related Information

UUIDValue Interface [page 253]

1.16.21 isNull Method

Tests if the value at the specified column name is null.

Overload list

Modifier and Type	Overload name	Description
public boolean	isNull(String) [page 156]	Tests if the value at the specified column name is null.
public boolean	isNull(int) [page 156]	Tests if the value at the specified column number is null.

In this section:

isNull(String) Method [page 156]

Tests if the value at the specified column name is null.

isNull(int) Method [page 156]
Tests if the value at the specified column number is null.

1.16.21.1 isNull(String) Method

Tests if the value at the specified column name is null.

Syntax

```
public boolean isNull (String name) throws ULjException
```

Parameters

name A String representing the table column name.

Returns

True if the value is null; otherwise, returns false.

1.16.21.2 isNull(int) Method

Tests if the value at the specified column number is null.

Syntax

```
public boolean isNull (int ordinal) throws ULjException
```

Parameters

ordinal A base-one integer representing the column number as ordered in the SQL statement.

Returns

True if the value is null; otherwise, returns false.

1.16.22 last() Method

Moves the cursor to the last row.

Syntax

```
public boolean last () throws ULjException
```

Returns

True on success; otherwise, returns false.

1.16.23 next() Method

Fetches the next row of data in the ResultSet object.

Syntax

```
public boolean next () throws ULjException
```

Returns

True when the next row is successfully fetched; otherwise, returns false.

Related Information

ResultSetMetadata Interface [page 158]

1.16.24 previous() Method

Fetches the previous row of data in the ResultSet object.

Syntax

```
public boolean previous () throws ULjException
```

Returns

True when the previous row is successfully fetched; otherwise, returns false.

1.16.25 relative(int) Method

Moves the cursor by an offset of rows from the current cursor position.

Syntax

```
public boolean relative (int offset) throws ULjException
```

Parameters

offset The number of rows to move.

Returns

True on success; otherwise, returns false.

1.17 ResultSetMetadata Interface

Associated with a ResultSet object and contains a method that provides column information.

Syntax

```
public interface ResultSetMetadata
```

Members

All members of ResultSetMetadata, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getAliasName(int) [page 161]	Returns the alias name for a column.
public int	getColumnCount() [page 161]	Returns the total number of columns in the ResultSet object.
public String	getCorrelationName(int) [page 162]	Returns the correlation name for a col-umn.
public String	getDomainName(int) [page 162]	Returns the name of the domain.
public int	getDomainPrecision(int) [page 163]	Returns the precision of the domain value.
public int	getDomainScale(int) [page 163]	Returns the scale of the domain value.
public int	getDomainSize(int) [page 164]	Returns the size of the domain value.
public short	getDomainType(int) [page 164]	Returns the type of the domain.
public String	getQualifiedName(int) [page 165]	Returns the qualified name for a col-umn.
public String	getTableColumnName(int) [page 165]	Returns the column name in the table or the derived table.
public String	getTableName(int) [page 166]	Returns the table name for a column.
public String	getWrittenName(int) [page 166]	Returns the written name for a column.

Remarks

This interface is obtained with the ResultSet.getResultSetMetadata method.

When a column in the select list of the ResultSet object is a simple name or a compound name (table-name.column-name, correlation-name.column-name), then, the following information about that name can be extracted if it exists:

- Alias name
- Correlation name
- Qualified version of the name
- Table name
- The name as written

For each column in the select list of the ResultSet object, the following information about the column domain can be obtained:

- Column Type: an integer from the Domain interface
- Name of the domain

- Size of the domain, for VARCHAR and BINARY domains
- Scale and precision, for NUMERIC domains

In this section:

Related Information

Domain Interface [page 72]
ResultSet Interface [page 129]
getResultSetMetadata() Method [page 149]

1.17.1 getAliasName(int) Method

Returns the alias name for a column.

Syntax

```
public String getAliasName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

Null if there is not an alias name for the column; otherwise, returns the alias name for a column.

Remarks

The alias name ([AS] name) may be specified to reference a column.

1.17.2 getColumnCount() Method

Returns the total number of columns in the ResultSet object.

Syntax

```
public int getColumnCount () throws ULjException
```

Returns

The number of columns.

1.17.3 getCorrelationName(int) Method

Returns the correlation name for a column.

Syntax

```
public String getCorrelationName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

Null if there is no correlation name for the column; otherwise, returns the correlation name for a column.

Remarks

The correlation name specified ([AS] correlation-name) in the FROM clause to specify a table expression, such as a derived table.

1.17.4 getDomainName(int) Method

Returns the name of the domain.

Syntax

```
public String getDomainName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

The domain name.

1.17.5 getDomainPrecision(int) Method

Returns the precision of the domain value.

Syntax

```
public int getDomainPrecision (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

The precision.

1.17.6 getDomainScale(int) Method

Returns the scale of the domain value.

Syntax

```
public int getDomainScale (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

The scale.

1.17.7 getDomainSize(int) Method

Returns the size of the domain value.

Syntax

```
public int getDomainSize (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

The size.

1.17.8 getDomainType(int) Method

Returns the type of the domain.

Syntax

```
public short getDomainType (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

The domain type expressed as an integer.

1.17.9 getQualifiedName(int) Method

Returns the qualified name for a column.

Syntax

```
public String getQualifiedName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

Null if there is no qualified name for the column; otherwise, returns the qualified name for a column.

Remarks

When the ResultSet column refers to a column in a table, the name returned is a compound name consisting of a correlation name (or table name if the correlation name was not given) followed by the name of the column in the table.

When the ResultSet column does not refer to a column in a table and an alias was specified, the alias name is returned.

1.17.10 getTableColumnName(int) Method

Returns the column name in the table or the derived table.

Syntax

```
public String getTableColumnName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

Null if there is no table name for the column; otherwise, returns the table name for a column.

1.17.11 getTableName(int) Method

Returns the table name for a column.

Syntax

```
public String getTableName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

Null if there is no table name for the column; otherwise, returns the table name for a column.

Remarks

The table is the name of the table that ResultSet column references (possibly as through a correlation name).

1.17.12 getWrittenName(int) Method

Returns the written name for a column.

Syntax

```
public String getWrittenName (int column_no) throws ULjException
```

Parameters

column_no The (base 1) number of the column in the select list.

Returns

Null if there is no written name for the column; otherwise, returns the written name for a column.

Remarks

The written name is the simple or compound name specified as the indicated column in the select list.

1.18 SQLInfo Interface

Represents information about an executed SQL statement.

Syntax

```
public interface SQLInfo
```

Members

All members of SQLInfo, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getMessage() [page 168]	Returns the message associated with the SQL code.
public String	getParameter(short) [page 168]	Returns the specified parameter.
public short	getParameterCount() [page 169]	Returns the number of parameters of the message.
public int	getSQLCode() [page 169]	Gets the code (SQLCODE) for the executed SQL statement.
public int	getSQLCount() [page 169]	Returns a value that depends on the result of the executed SQL statement.

In this section:

1.18.1 getMessage() Method

Returns the message associated with the SQL code.

Syntax

```
public String getMessage ()
```

1.18.2 getParameter(short) Method

Returns the specified parameter.

Syntax

```
public String getParameter (short param_no)
```

Parameters

param_no A one-based parameter number.

Returns

The parameter.

1.18.3 getParameterCount() Method

Returns the number of parameters of the message.

Syntax

```
public short getParameterCount ()
```

Returns

The number of parameters.

1.18.4 getSQLCode() Method

Gets the code (SQLCODE) for the executed SQL statement.

Syntax

```
public int getSQLCode ()
```

Returns

The SQLCODE value.

1.18.5 getSQLCount() Method

Returns a value that depends on the result of the executed SQL statement.

Syntax

```
public int getSQLCount ()
```

Returns

The number of rows affected by the statement after an INSERT, UPDATE, or DELETE statement is executed. The return value is the offset into the associated dynamic SQL statement that corresponds to the error if a SQLE_SYNTAX_ERROR occurs.

1.19 StreamHTTPParms Interface

Represents HTTP stream parameters that define how to communicate with a MobiLink server using HTTP.

Syntax

```
public interface StreamHTTPParms extends StreamTCPIPParms
```

Members

All members of StreamHTTPParms, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getE2eePublicKey() [page 173]	Returns the name of the file containing the end-to-end public key.
public String	getExtraParameters() [page 173]	Gets the extra MobiLink client network protocol options.
public String	getHost() [page 173]	Returns the host name of the MobiLink server.
public int	getOutputBufferSize() [page 174]	Returns the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.
public int	getPort() [page 175]	Returns the port number used to connect to the MobiLink server.
public String	getURLSuffix() [page 175]	Returns the URL suffix of the MobiLink server.
public boolean	isRestartable() [page 176]	Determines whether restartable HTTP is used.
public void	setE2eePublicKey(String) [page 176]	Specifies the name of the file containing the end-to-end public key.
public void	setExtraParameters(String) [page 177]	Sets extra MobiLink client network protocol options.

Modifier and Type	Method	Description
public void	setHost(String) [page 178]	Sets the host name of the MobiLink server.
public void	setOutputBufferSize(int) [page 178]	Sets the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.
public void	setPort(int) [page 179]	Sets the port number used to connect to the MobiLink server.
public void	setRestartable(boolean) [page 180]	Enables or disables restartable HTTP.
public void	setURLSuffix(String) [page 180]	Specifies the URL suffix to connect to the MobiLink server.
public void	setZlibCompression(boolean) [page 181]	Enables or disables ZLIB compression.
public void	setZlibDownloadWindowSize(int) [page 181]	Sets the download window size for ZLIB compression.
public void	setZlibUploadWindowSize(int) [page 182]	Sets the upload window size for ZLIB compression.
public boolean	zlibCompressionEnabled() [page 182]	Determines if ZLIB compression is enabled.

Remarks

The following example sets the stream parameters to communicate with a MobiLink server on host name "MyMLHost". The server started with the following parameters: "-x http(port=1234)":

```
SyncParms syncParms = myConnection.createSyncParms(
   SyncParms.HTTP_STREAM,
   "MyUniqueMLUserID",
   "MyMLScriptVersion"
);
StreamHTTPParms httpParms = syncParms.getStreamParms();
httpParms.setHost("MyMLHost");
httpParms.setPort(1234);
```

Instances implementing this interface are returned by the SyncParms.getStreamParms method.

In this section:

getPort() Method [page 175]
    Returns the port number used to connect to the MobiLink server.

getURLSuffix() Method [page 175]
    Returns the URL suffix of the MobiLink server.

isRestartable() Method [page 176]
    Determines whether restartable HTTP is used.

setE2eePublicKey(String) Method [page 176]
    Specifies the name of the file containing the end-to-end public key.

setExtraParameters(String) Method [page 177]
    Sets extra MobiLink client network protocol options.

setHost(String) Method [page 178]
    Sets the host name of the MobiLink server.

setOutputBufferSize(int) Method [page 178]
    Sets the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.

setPort(int) Method [page 179]
    Sets the port number used to connect to the MobiLink server.

setRestartable(boolean) Method [page 180]
    Enables or disables restartable HTTP.

setURLSuffix(String) Method [page 180]
    Specifies the URL suffix to connect to the MobiLink server.

setZlibCompression(boolean) Method [page 181]
    Enables or disables ZLIB compression.

setZlibDownloadWindowSize(int) Method [page 181]
    Sets the download window size for ZLIB compression.

setZlibUploadWindowSize(int) Method [page 182]
    Sets the upload window size for ZLIB compression.

zlibCompressionEnabled() Method [page 182]
    Determines if ZLIB compression is enabled.

Related Information

SyncParms Class [page 197]
getStreamParms() Method [page 206]

1.19.1 getE2eePublicKey() Method

Returns the name of the file containing the end-to-end public key.

Syntax

```
public String getE2eePublicKey ()
```

Returns

The name of the file containing the end-to-end public key.

Related Information

setE2eePublicKey(String) Method [page 176]

1.19.2 getExtraParameters() Method

Gets the extra MobiLink client network protocol options.

Syntax

```
public String getExtraParameters ()
```

Returns

The extra protocol options that have been set.

1.19.3 getHost() Method

Returns the host name of the MobiLink server.

Syntax

```
public String getHost ()
```

Returns

The name of the host.

Related Information

setHost(String) Method [page 178]
getPort() Method [page 175]
setPort(int) Method [page 179]

1.19.4 getOutputBufferSize() Method

Returns the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.

Syntax

```
public int getOutputBufferSize ()
```

Returns

The integer containing the buffer size.

Remarks

Increasing this value may reduce the number of network flushes needed to send a large upload at the cost of increased memory use. In HTTP, each flush sends a large (approximately 250 bytes) HTTP header; reducing the number of flushes can reduce the bandwidth use.

Related Information

setOutputBufferSize(int) Method [page 178]

1.19.5 getPort() Method

Returns the port number used to connect to the MobiLink server.

Syntax

```
public int getPort ()
```

Returns

The port number of MobiLink server.

Related Information

setPort(int) Method [page 179]

1.19.6 getURLSuffix() Method

Returns the URL suffix of the MobiLink server.

Syntax

```
public String getURLSuffix ()
```

Returns

The String containing the URL suffix.

Related Information

setURLSuffix(String) Method [page 180]

1.19.7 isRestartable() Method

Determines whether restartable HTTP is used.

Syntax

```
public boolean isRestartable ()
```

Returns

True if restartable HTTP is enabled; otherwise, returns false.

Related Information

setRestartable(boolean) Method [page 180]

1.19.8 setE2eePublicKey(String) Method

Specifies the name of the file containing the end-to-end public key.

Syntax

```
public void setE2eePublicKey (String public_key)
```

Parameters

public_key The name of the RSA public key file used in the encryption. This name must be DER-encoded.

Remarks

By default, this value is null, indicating that end-to-end encryption is not used.

This method corresponds to the e2ee_public_key protocol option.

The public key can be stored on either an SD card or the device object store.

When using an SD card, the public_key parameter should have the following form:

file:// path

path is the absolute path to the file on the card. For example, `file:///SDCard/ulj/public_key.der` is a valid public_key parameter.

When using the object store, use the UltraLite Java Edition Database Transfer utility to download the file from the MobiLink server. The key must have a `der` file extension.

Related Information

getE2eePublicKey() Method [page 173]

1.19.9 setExtraParameters(String) Method

Sets extra MobiLink client network protocol options.

Syntax

```
public void setExtraParameters (String parms)
```

Parameters

parms A semicolon delimited list of protocol options.

Remarks

These options are appended to the list that is built from the settings resulting from the methods of this class.

Options that are set by this method override the same options that are set by other methods. For example, if "host=abc" is contained in the extra parameters, and the setHost("xyz") method is called, then the host option is "abc".

1.19.10 setHost(String) Method

Sets the host name of the MobiLink server.

Syntax

```
public void setHost (String v)
```

Parameters

v The name of the host.

Remarks

The default is null, which indicates a localhost.

Related Information

getHost() Method [page 173]
getPort() Method [page 175]
setPort(int) Method [page 179]

1.19.11 setOutputBufferSize(int) Method

Sets the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.

Syntax

```
public void setOutputBufferSize (int size)
```

Parameters

size The new buffer size.

Remarks

The default is 4096. Valid values range between 512 and 32768. Increasing this value may cause the Java runtime to send chunked HTTP, which the MobiLink server cannot process.

If the MobiLink server outputs an "unknown transfer encoding" error, try decreasing this value.

Related Information

getOutputBufferSize() Method [page 174]

1.19.12 setPort(int) Method

Sets the port number used to connect to the MobiLink server.

Syntax

```
public void setPort (int v)
```

Parameters

v A port number ranging from 1 to 65535. Out of range values revert to default value.

Remarks

The default port is 80 for HTTP synchronizations and 443 for HTTPS synchronizations.

Related Information

getPort() Method [page 175]

1.19.13 setRestartable(boolean) Method

Enables or disables restartable HTTP.

Syntax

```
public void setRestartable (boolean isRestartable)
```

Parameters

isRestartable Set to true to enable restartable HTTP. The default value is false.

Remarks

When restartable HTTP is enabled, UltraLiteJ can tolerate network interruptions so that synchronizations do not fail as often on unreliable networks.

To use restartable HTTP, both UltraLiteJ and the MobiLink server must have applied CR#690250.

Related Information

isRestartable() Method [page 176]

1.19.14 setURLSuffix(String) Method

Specifies the URL suffix to connect to the MobiLink server.

Syntax

```
public void setURLSuffix (String v)
```

Parameters

v The URL suffix string.

Remarks

UltraLiteJ forms URLs in the following format:

```
[http|https]://host-name:port-number/url-suffix
```

By default, *url-suffix* is "Mobilink/". You can set the URL suffix to the default by setting **v** to null.

Related Information

getURLSuffix() Method [page 175]

1.19.15 setZlibCompression(boolean) Method

Enables or disables ZLIB compression.

Syntax

```
public void setZlibCompression (boolean enable)
```

Parameters

enable Set to true to enable ZLIB compression, or false to disable ZLIB compression.

Remarks

By default, ZLIB compression is disabled.

This method corresponds to the compression=zlib protocol option.

1.19.16 setZlibDownloadWindowSize(int) Method

Sets the download window size for ZLIB compression.

Syntax

```
public void setZlibDownloadWindowSize (int size)
```

Parameters

size The compression window size specification. This parameter is the base two logarithm of the window size (the size of the history buffer). Specify a valid range from 9 to 15, inclusively.

Remarks

This method corresponds to the zlib_download_window_size protocol option.

1.19.17 setZlibUploadWindowSize(int) Method

Sets the upload window size for ZLIB compression.

Syntax

```
public void setZlibUploadWindowSize (int size)
```

Parameters

size The compression window size specification. This parameter is the base two logarithm of the window size (the size of the history buffer). Specify a valid range from 9 to 15, inclusively.

Remarks

This method corresponds to the zlib_upload_window_size protocol option.

1.19.18 zlibCompressionEnabled() Method

Determines if ZLIB compression is enabled.

Syntax

```
public boolean zlibCompressionEnabled ()
```

Returns

True if enabled; otherwise, returns false.

Related Information

setZlibCompression(boolean) Method [page 181]

1.20 StreamHTTPSParms Interface

Represents HTTPS stream parameters that define how to communicate with a MobiLink server using secure HTTPS connections.

Syntax

```
public interface StreamHTTPSParms extends StreamHTTPParms
```

Members

All members of StreamHTTPSParms, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getCertificateCompany() [page 186]	Returns the certificate company name for verification of secure connections.
public String	getCertificateName() [page 186]	Returns the certificate common name for verification of secure connections.
public String	getCertificateUnit() [page 187]	Returns the certificate unit name for verification of secure connections.
public String	getTrustedCertificates() [page 187]	Returns the name of the file containing a list of trusted root certificates used for secure synchronization.
public void	setCertificateCompany(String) [page 188]	Sets the certificate company name for verification of secure connections.
public void	setCertificateName(String) [page 188]	Sets the certificate common name for verification of secure connections.
public void	setCertificateUnit(String) [page 189]	Sets the certificate unit name for verifi-cation of secure connections.

Modifier and Type	Method	Description
public void	setTrustedCertificates(String) [page 189]	Sets a file containing a list of trusted root certificates used for secure synchronization.

Inherited members from StreamHTTPParms

Modifier and Type	Member	Description
public String	getE2eePublicKey() [page 173]	Returns the name of the file containing the end-to-end public key.
public String	getExtraParameters() [page 173]	Gets the extra MobiLink client network protocol options.
public String	getHost() [page 173]	Returns the host name of the MobiLink server.
public int	getOutputBufferSize() [page 174]	Returns the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.
public int	getPort() [page 175]	Returns the port number used to connect to the MobiLink server.
public String	getURLSuffix() [page 175]	Returns the URL suffix of the MobiLink server.
public boolean	isRestartable() [page 176]	Determines whether restartable HTTP is used.
public void	setE2eePublicKey(String) [page 176]	Specifies the name of the file containing the end-to-end public key.
public void	setExtraParameters(String) [page 177]	Sets extra MobiLink client network protocol options.
public void	setHost(String) [page 178]	Sets the host name of the MobiLink server.
public void	setOutputBufferSize(int) [page 178]	Sets the size, in bytes, of the output buffer used to store data before it is sent to the MobiLink server.
public void	setPort(int) [page 179]	Sets the port number used to connect to the MobiLink server.
public void	setRestartable(boolean) [page 180]	Enables or disables restartable HTTP.
public void	setURLSuffix(String) [page 180]	Specifies the URL suffix to connect to the MobiLink server.
public void	setZlibCompression(boolean) [page 181]	Enables or disables ZLIB compression.
public void	setZlibDownloadWindowSize(int) [page 181]	Sets the download window size for ZLIB compression.
public void	setZlibUploadWindowSize(int) [page 182]	Sets the upload window size for ZLIB compression.

Modifier and Type	Member	Description
public boolean	zlibCompressionEnabled() [page 182]	Determines if ZLIB compression is enabled.

Remarks

The following example sets the stream parameters to communicate with a MobiLink server on host name "MyMLHost". The server started with the following parameters: "-x https(port=1234;identity=RSAServer.id;identity_password=x)"

```
SyncParms syncParms = myConnection.createSyncParms(
        SyncParms.HTTPS_STREAM,
        "MyUniqueMLUserID",
        "MyMLScriptVersion"
    );
StreamHTTPSParms httpsParms =
    (StreamHTTPSParms) syncParms.getStreamParms();
httpsParms.setHost("MyMLHost");
httpsParms.setPort(1234);
```

The above example assumes that the identity in RSAServer.id is chained to a trusted root identity already installed on the client host or device.

For J2SE, you can deploy the required trusted root identity using one of the following methods:

1. Install the trusted root identity in the lib/security/cacerts key store of the JRE.
2. Build your own key store using the Java keytool utility and setting the javax.net.ssl.trustStore Java system property to its location (set the javax.net.ssl.trustStorePassword method to an appropriate value)
3. Use the setTrustedCertificates(String) parameter to point to the deployed identity file.

To enhance security, the setCertificateName, setCertificateCompany, and setCertificateUnit methods should be used to turn on validation of the MobiLink server identity.

Instances implementing this interface are returned by the SyncParms.getStreamParms method when the SyncParms object is created for HTTPS synchronization.

In this section:

setCertificateName(String) Method [page 188]
: Sets the certificate common name for verification of secure connections.

setCertificateUnit(String) Method [page 189]
: Sets the certificate unit name for verification of secure connections.

setTrustedCertificates(String) Method [page 189]
: Sets a file containing a list of trusted root certificates used for secure synchronization.

Related Information

SyncParms Class [page 197]
getStreamParms() Method [page 206]
setCertificateCompany(String) Method [page 188]
setCertificateName(String) Method [page 188]
setCertificateUnit(String) Method [page 189]
setTrustedCertificates(String) Method [page 189]

1.20.1 getCertificateCompany() Method

Returns the certificate company name for verification of secure connections.

Syntax

```
public String getCertificateCompany ()
```

Returns

The certificate company name.

1.20.2 getCertificateName() Method

Returns the certificate common name for verification of secure connections.

Syntax

```
public String getCertificateName ()
```

Returns

The certificate name.

1.20.3 getCertificateUnit() Method

Returns the certificate unit name for verification of secure connections.

Syntax

```
public String getCertificateUnit ()
```

Returns

The organization unit name.

1.20.4 getTrustedCertificates() Method

Returns the name of the file containing a list of trusted root certificates used for secure synchronization.

Syntax

```
public String getTrustedCertificates ()
```

Returns

The file name of the trusted root certificates file.

Related Information

setTrustedCertificates(String) Method [page 189]

1.20.5 setCertificateCompany(String) Method

Sets the certificate company name for verification of secure connections.

Syntax

```
public void setCertificateCompany (String val)
```

Parameters

val The company name.

Remarks

The default is null, indicating that the company name does not get verified in the certificate.

1.20.6 setCertificateName(String) Method

Sets the certificate common name for verification of secure connections.

Syntax

```
public void setCertificateName (String val)
```

Parameters

val The certificate common name.

Remarks

The default is null, indicating that the common name does not get verified in the certificate.

1.20.7 setCertificateUnit(String) Method

Sets the certificate unit name for verification of secure connections.

Syntax

```
public void setCertificateUnit (String val)
```

Parameters

val The company unit name.

Remarks

The default is null, indicating that the organization unit name does not get verified in the certificate.

1.20.8 setTrustedCertificates(String) Method

Sets a file containing a list of trusted root certificates used for secure synchronization.

Syntax

```
public void setTrustedCertificates (String filename) throws ULjException
```

Parameters

filename The file name of the trusted root certificate.

Remarks

This method supports any X.509 format that the Java Runtime Environment on the platform allows. The BKS KeyStore is used.

Certificates are used according to the following rules of precedence:

1. If this method is called, then the certificates from the specified file are used.

2. If this method is not called and certificates were set in the database by the ulinit or ulload utilities, then those certificates are used.
3. If certificates are not specified by either this method or by the ulinit or ulload utilities, then certificates are read from the operating system's trusted certificate store. This certificate store is used by web browsers when they connect to secure web servers via HTTPS.

Related Information

getTrustedCertificates() Method [page 187]

1.21 StreamTCPIPParms Interface

Represents TCP/IP stream parameters that define how to communicate with a MobiLink server using TCP/IP.

Syntax

```
public interface StreamTCPIPParms
```

Members

All members of StreamHTTPSParms, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getE2eePublicKey() [page 173]	Returns the name of the file containing the end-to-end public key.
public String	getHost() [page 173]	Returns the host name of the MobiLink server.
public String	getPort() [page 175]	Returns the port number used to connect to the MobiLink server.
public void	setE2eePublicKey(String) [page 176]	Specifies the name of the file containing the end-to-end public key.
public void	setExtraParameters(String) [page 177]	Sets extra MobiLink client network protocol options.
public void	setHost(String) [page 178]	Sets the host name of the MobiLink server.
public void	setPort(int) [page 179]	Sets the port number used to connect to the MobiLink server.

Modifier and Type	Method	Description
public void	setZlibCompression(boolean) [page 181]	Enables or disables ZLIB compression.
public void	setZlibDownloadWindowSize(int) [page 181]	Sets the download window size for ZLIB compression.
public void	setZlibUploadWindowSize(int) [page 182]	Sets the upload window size for ZLIB compression.
public void	zlibCompressionEnabled() method [page 182]	Determines if ZLIB compression is enabled.

Remarks

The following example sets the stream parameters to communicate with a MobiLink server on host name "MyMLHost." The server starts with the following parameters: "-x tcpip(port=1234)":

```
SyncParms syncParms = myConnection.createSyncParms(
    SyncParms.TCPIP_STREAM,
    "MyUniqueMLUserID",
    "MyMLScriptVersion"
);
StreamTCPIPParms sparms = syncParms.getStreamParms();
sparms.setHost("MyMLHost");
sParms.setPort(1234);
```

Instances implementing this interface are returned by the SyncParms.getStreamParms method.

1.22 StreamTLSParms Interface

Represents TLS stream parameters that define how to communicate with a MobiLink server using secure TLS connections.

Syntax

```
public interface StreamTLSParms
```

Members

All members of StreamHTTPSParms, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getCertificateCompany() [page 186]	Returns the certificate company name for verification of secure connections.
public String	getCertificateName() [page 186]	Returns the certificate common name for verification of secure connections.
public String	getCertificateUnit() [page 187]	Returns the certificate unit name for verification of secure connections.
public String	getTrustedCertificates() [page 187]	Returns the name of the file containing a list of trusted root certificates used for secure synchronization.
public void	setCertificateCompany(String) [page 188]	Sets the certificate company name for verification of secure connections.
public void	setCertificateName(String) [page 188]	Sets the certificate common name for verification of secure connections.
public void	setCertificateUnit(String) [page 189]	Sets the certificate unit name for verification of secure connections.
public void	setTrustedCertificates(String) [page 189]	Sets a file containing a list of trusted root certificates used for secure synchronization.

Remarks

The following example sets the stream parameters to communicate with a MobiLink server on host name "MyMLHost". The server started with the following parameters: "-x tls(port=1234;identity=RSAServer.id;identity_password=x)":

```
SyncParms syncParms = myConnection.createSyncParms(
        SyncParms.TLS_STREAM,
        "MyUniqueMLUserID",
        "MyMLScriptVersion"
    );
StreamTLSParms sParms =
    (StreamTLSParms) syncParms.getStreamParms();
sParms.setHost("MyMLHost");
sParms.setPort(1234);
```

The above example assumes that the identity in RSAServer.id is chained to a trusted root identity already installed on the client host or device.

For J2SE, you can deploy the required trusted root identity using one of the following methods:

1. Install the trusted root identity in the lib/security/cacerts key store of the JRE.
2. Build your own key store using the Java keytool utility and setting the javax.net.ssl.trustStore Java system property to its location (set the javax.net.ssl.trustStorePassword method to an appropriate value).
3. Use the setTrustedCertificates(String) parameter to point to the deployed identity file.

To enhance security, the setCertificateName, setCertificateCompany, and setCertificateUnit methods should be used to turn on validation of the MobiLink server identity.

Instances implementing this interface are returned by the SyncParms.getStreamParms method when the SyncParms object is created for TLS synchronization.

1.23 SyncObserver Interface

Receives synchronization progress information.

Syntax

```
public interface SyncObserver
```

Members

All members of SyncObserver, including inherited members.

Methods

Modifier and Type	Method	Description
public boolean	syncProgress(int, SyncResult) [page 194]	Informs the user of progress.

Remarks

Create a new class that performs synchronization, and implement it using the SyncParms.setSyncObserver method to receive synchronization progress reports.

The following example illustrates a simple SyncObserver object implementation:

```
class MyObserver implements SyncObserver {
 public boolean syncProgress(int state, SyncResult result) {
   System.out.println(
       "sync progress state = " + state
       + " bytes sent = " + result.getSentByteCount()
       + " bytes received = " + result.getReceivedByteCount()
     );
   return false;   // Always continue synchronization.
 }
 public MyObserver() {} // The default constructor.
}
```

The above class can be enabled with the following method call:

```
SyncParms.setSyncObserver(new MyObserver());
```

In this section:

syncProgress(int, SyncResult) Method [page 194]
    Informs the user of progress.

Related Information

SyncParms Class [page 197]
setSyncObserver(SyncObserver) Method [page 220]

1.23.1 syncProgress(int, SyncResult) Method

Informs the user of progress.

Syntax

```
public boolean syncProgress (
    int state,
    SyncResult data
)
```

Parameters

state One of the SyncObserver.States constants, representing the current state of the synchronization.
data A SyncResult object containing the latest synchronization results.

Returns

return True to cancel the synchronization; otherwise, returns false to continue synchronization.

Remarks

This method is invoked during synchronization.

The various states, which are signaled as packets, are received and sent. Since multiple tables may be uploaded or downloaded in a single packet, calls to this method for any given synchronization may skip a number of states.

i Note

With the exception of the SyncResult methods, no other UltraLiteJ API methods should be invoked during a syncProgress call.

Related Information

SyncObserver.States Interface [page 195]
setSyncObserver(SyncObserver) Method [page 220]
SyncResult Class [page 224]

1.24 SyncObserver.States Interface

Defines the synchronization states that can be signaled to an observer.

Syntax

```
public interface States
```

Members

All members of States, including inherited members.

Variables

Modifier and Type	Variable	Description
public final int	STARTING	Denotes that a synchronization is starting. No actions have taken place yet.
public final int	CONNECTING	Denotes that a synchronization is starting. No actions have taken place yet.

Modifier and Type	Variable	Description
public final int	RESUMING_DOWNLOAD	An optional state that is entered when we are attempting to resume a partial download. On success, sync will proceed to the UL_SYNC_STATE_RECEIVING_TABLE state, and UL_SYNC_STATE_ERROR if we are unable to resume.
public final int	SENDING_HEADER	Denotes that the synchronization stream has been opened and the header is about to be sent.
public final int	SENDING_CHECK_SYNC_REQUEST	The state of the last upload is unknown, so a request to check its status is being sent.
public final int	WAITING_FOR_CHECK_SYNC_RE-SPONSE	Waiting for the server to respond to the check sync request.
public final int	PROCESSING_CHECK_SYNC_RE-SPONSE	The response to the check sync request has been received and is being proc-essed.
public final int	SENDING_TABLE	Denotes that a new table is being up-loaded.
public final int	SENDING_DATA	Denotes that schema information or row data is being sent.
public final int	FINISHING_UPLOAD	Denotes that the upload is finalizing.
public final int	WAITING_FOR_UPLOAD_ACK	Waiting for the server to acknowledge receiving our upload.
public final int	PROCESSING_UPLOAD_ACK	The server has acknowledged receiving our upload.
public final int	WAITING_FOR_DOWNLOAD	Waiting for the server to start sending the download.
public final int	RECEIVING_TABLE	Denotes that a new table is being down-loaded.
public final int	RECEIVING_DATA	Denotes that schema information or row data is being received.
public final int	COMMITTING_DOWNLOAD	Denotes that the downloaded rows are being committed to the database.
public final int	ROLLING_BACK_DOWNLOAD	Denotes that the downloaded rows are being committed to the database.
public final int	SENDING_DOWNLOAD_ACK	Denotes that an acknowledgement of a complete download is being sent.
public final int	DISCONNECTING	Denotes that the synchronization stream is disconnecting.

Modifier and Type	Variable	Description
public final int	DONE	Denotes that synchronization is complete. No other states are reported.
public final int	ERROR	Denotes that synchronization is complete but an error occurred.

Related Information

setSyncObserver(SyncObserver) Method [page 220]
SyncObserver Interface [page 193]

1.25 SyncParms Class

Maintains the parameters used during the database synchronization process.

Syntax

```
public abstract class SyncParms
```

Members

All members of SyncParms, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final int	HTTP_STREAM	Creates a SyncParms object for HTTP synchronizations.
public static final int	HTTPS_STREAM	Creates a SyncParms object for secure HTTPS synchronizations.
public static final int	TCPIP_STREAM	Creates a SyncParms object for TCP/IP synchronizations.
public static final int	TLS_STREAM	Creates a SyncParms object for TLS synchronizations.

Methods

Modifier and Type	Method	Description
public abstract boolean	getAcknowledgeDownload() [page 202]	Determines if the client sends download acknowledgments.
public abstract String	getAdditionalParms() [page 202]	Returns the additional synchronization parameters.
public abstract String	getAuthenticationParms() [page 203]	Returns parameters provided to a custom user authentication script.
public abstract boolean	getKeepPartialDownload() [page 203]	Determines if partial downloads are enabled.
public abstract int	getLivenessTimeout() [page 204]	Returns the liveness timeout length, in seconds.
public abstract String	getNewPassword() [page 204]	Returns the new MobiLink password for the user specified with the setUserName method.
public abstract String	getPassword() [page 205]	Returns the MobiLink password for the user specified with the setUserName method.
public abstract String	getPublications() [page 205]	Returns the publications to be synchronized.
public abstract boolean	getResumePartialDownload() [page 206]	Determines if partial downloads are to be resumed.
public abstract StreamHTTPParms	getStreamParms() [page 206]	Returns the parameters used to configure the synchronization stream.
public abstract SyncObserver	getSyncObserver() [page 207]	Returns the currently specified SyncObserver object.
public abstract SyncResult	getSyncResult() [page 208]	Returns the SyncResult object that contains the status of the synchronization.
public abstract String	getTableOrder() [page 209]	Returns the order in which tables should be uploaded to the consolidated database.
public abstract String	getUserName() [page 209]	Returns the MobiLink user name that uniquely identifies the client to the MobiLink server.
public abstract String	getVersion() [page 210]	Returns the script version to use.
public abstract boolean	isDownloadOnly() [page 210]	Determines if the synchronization is download-only.
public abstract boolean	isPingOnly() [page 211]	Determines whether the client pings the MobiLink server or performs a synchronization.
public abstract boolean	isUploadOnly() [page 211]	Determines if the synchronization is upload-only.
public abstract void	setAcknowledgeDownload(boolean) [page 212]	Specifies whether the client should send download acknowledgements.

Modifier and Type	Method	Description
public abstract void	setAdditionalParms(String) [page 212]	Specifies additional synchronization parameters as a semicolon-separated list of name=value pairs.
public abstract void	setAuthenticationParms(String) [page 213]	Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).
public abstract void	setDownloadOnly(boolean) [page 214]	Sets the synchronization as download-only.
public abstract void	setKeepPartialDownload(boolean) [page 214]	Specifies whether partial downloads should be allowed while synchronizing.
public abstract void	setLivenessTimeout(int) [page 216]	Sets the liveness timeout length, in seconds.
public abstract void	setNewPassword(String) [page 216]	Sets a new MobiLink password for the user specified with setUserName method.
public abstract void	setPassword(String) [page 217]	Sets the MobiLink password for the user specified with the setUserName method.
public abstract void	setPingOnly(boolean) [page 218]	Sets the client to ping the MobiLink server rather than perform a synchronization.
public abstract void	setPublications(String) [page 218]	Sets the publications to be synchronized.
public abstract void	setResumePartialDownload(boolean) [page 219]	Specifies whether to resume or discard a previous partial download.
public abstract void	setSyncObserver(SyncObserver) [page 220]	Sets a SyncObserver object to monitor the progress of the synchronization.
public abstract void	setTableOrder(String) [page 221]	Sets the order in which tables should be uploaded to the consolidated database.
public abstract void	setUploadOnly(boolean) [page 221]	Sets the synchronization as upload-only.
public abstract void	setUserName(String) [page 222]	Sets the MobiLink user name that uniquely identifies the client to the MobiLink server.
public abstract void	setVersion(String) [page 223]	Sets the synchronization script to use.

Remarks

This interface is invoked with the Connection.createSyncParms method.

You can only set one synchronization command at a time. These commands are specified using the setDownloadOnly, setPingOnly, and setUploadOnly methods. By setting one of these methods to true, you set the other methods to false.

The UserName and Version parameters must be set. The UserName must be unique for each client database.

The communication stream is configured using the getStreamParms method based on the type of SyncParms object. For example, the following code prepares and performs an HTTP synchronization:

```
SyncParms syncParms = myConnection.createSyncParms(
   SyncParms.HTTP_STREAM,
   "MyUniqueMLUserID",
   "MyMLScriptVersion"
);
syncParms.setPassword("ThePWDforMyUniqueMLUserID");
syncParms.getStreamParms().setHost("MyMLHost");
myConnection.synchronize(syncParms);
```

Comma Separated Lists

AuthenticationParms, Publications, and TableOrder parameters are all specified using a string value that contains a comma separated list of values. Values within the list may be quoted using either single quotes or double quotes, but there are no escape characters. Leading and trailing spaces in values are ignored unless quoted. For example, the following code specifies *Table A*, then *Table B,D*, then *Table C*:

```
syncParms.setTableOrder( "'Table A',\"Table B,D\",Table C );
```

In this section:

getTableOrder() Method [page 209]
Returns the order in which tables should be uploaded to the consolidated database.

getUserName() Method [page 209]
Returns the MobiLink user name that uniquely identifies the client to the MobiLink server.

getVersion() Method [page 210]
Returns the script version to use.

isDownloadOnly() Method [page 210]
Determines if the synchronization is download-only.

isPingOnly() Method [page 211]
Determines whether the client pings the MobiLink server or performs a synchronization.

isUploadOnly() Method [page 211]
Determines if the synchronization is upload-only.

setAcknowledgeDownload(boolean) Method [page 212]
Specifies whether the client should send download acknowledgements.

setAdditionalParms(String) Method [page 212]
Specifies additional synchronization parameters as a semicolon-separated list of name=value pairs.

setAuthenticationParms(String) Method [page 213]
Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

setDownloadOnly(boolean) Method [page 214]
Sets the synchronization as download-only.

setKeepPartialDownload(boolean) Method [page 214]
Specifies whether partial downloads should be allowed while synchronizing.

setLivenessTimeout(int) Method [page 216]
Sets the liveness timeout length, in seconds.

setNewPassword(String) Method [page 216]
Sets a new MobiLink password for the user specified with setUserName method.

setPassword(String) Method [page 217]
Sets the MobiLink password for the user specified with the setUserName method.

setPingOnly(boolean) Method [page 218]
Sets the client to ping the MobiLink server rather than perform a synchronization.

setPublications(String) Method [page 218]
Sets the publications to be synchronized.

setResumePartialDownload(boolean) Method [page 219]
Specifies whether to resume or discard a previous partial download.

setSyncObserver(SyncObserver) Method [page 220]
Sets a SyncObserver object to monitor the progress of the synchronization.

setTableOrder(String) Method [page 221]
Sets the order in which tables should be uploaded to the consolidated database.

setUploadOnly(boolean) Method [page 221]
Sets the synchronization as upload-only.

setUserName(String) Method [page 222]

Sets the MobiLink user name that uniquely identifies the client to the MobiLink server.

setVersion(String) Method [page 223]
Sets the synchronization script to use.

Related Information

getStreamParms() Method [page 206]
setUserName(String) Method [page 222]
createSyncParms(int, String, String) Method [page 43]
StreamHTTPParms Interface [page 170]
StreamHTTPSParms Interface [page 183]

1.25.1 getAcknowledgeDownload() Method

Determines if the client sends download acknowledgments.

Syntax

```
public abstract boolean getAcknowledgeDownload ()
```

Returns

True if the client sends download acknowledgments; otherwise, returns false.

Related Information

setAcknowledgeDownload(boolean) Method [page 212]

1.25.2 getAdditionalParms() Method

Returns the additional synchronization parameters.

Syntax

```
public abstract String getAdditionalParms ()
```

Returns

The list of additional parameters or null if no parameters are specified.

Related Information

setAdditionalParms(String) Method [page 212]

1.25.3 getAuthenticationParms() Method

Returns parameters provided to a custom user authentication script.

Syntax

```
public abstract String getAuthenticationParms ()
```

Returns

The list of authentication parms or null if no parameters are specified.

Related Information

setAuthenticationParms(String) Method [page 213]

1.25.4 getKeepPartialDownload() Method

Determines if partial downloads are enabled.

Syntax

```
public abstract boolean getKeepPartialDownload ()
```

Returns

True if partial downloads are enabled; otherwise, returns false.

Related Information

setKeepPartialDownload(boolean) Method [page 214]

1.25.5 getLivenessTimeout() Method

Returns the liveness timeout length, in seconds.

Syntax

```
public abstract int getLivenessTimeout ()
```

Returns

The timeout.

Related Information

setLivenessTimeout(int) Method [page 216]

1.25.6 getNewPassword() Method

Returns the new MobiLink password for the user specified with the setUserName method.

Syntax

```
public abstract String getNewPassword ()
```

Returns

The new password set after the next synchronization.

Related Information

setUserName(String) Method [page 222]
setNewPassword(String) Method [page 216]

1.25.7 getPassword() Method

Returns the MobiLink password for the user specified with the setUserName method.

Syntax

```
public abstract String getPassword ()
```

Returns

The password for the MobiLink user.

Related Information

setPassword(String) Method [page 217]

1.25.8 getPublications() Method

Returns the publications to be synchronized.

Syntax

```
public abstract String getPublications ()
```

Returns

The set of publications to synchronize.

Related Information

setPublications(String) Method [page 218]

1.25.9 getResumePartialDownload() Method

Determines if partial downloads are to be resumed.

Syntax

```
public abstract boolean getResumePartialDownload ()
```

Returns

True if partial downloads are to be resumed; otherwise, returns false.

Related Information

setResumePartialDownload(boolean) Method [page 219]

1.25.10 getStreamParms() Method

Returns the parameters used to configure the synchronization stream.

Syntax

```
public abstract StreamHTTPParms getStreamParms ()
```

Returns

A StreamHTTPParms or StreamHTTPSParms object specifying the parameters for HTTP or HTTPS synchronization streams. The object is returned by reference.

Remarks

The synchronization stream type is specified when the SyncParms object is created.

Related Information

createSyncParms(int, String, String) Method [page 43]
StreamHTTPParms Interface [page 170]
StreamHTTPSParms Interface [page 183]

1.25.11 getSyncObserver() Method

Returns the currently specified SyncObserver object.

Syntax

```
public abstract SyncObserver getSyncObserver ()
```

Returns

The SyncObserver object, or null if an observer was not specified.

Related Information

setSyncObserver(SyncObserver) Method [page 220]

1.25.12 getSyncResult() Method

Returns the SyncResult object that contains the status of the synchronization.

Syntax

```
public abstract SyncResult getSyncResult ()
```

Returns

The SyncResult object representing the result of the last call to the Connection.synchronize method.

Remarks

The following example illustrates how to get the result set of the last call to the Connection.synchronize method:

```
conn.synchronize( mySyncParms );
SyncResult result = mySyncParms.getSyncResult();
display(
   "*** Synchronized *** sent=" + result.getSentRowCount()
   + ", received=" + result.getReceivedRowCount()
);
```

Note

This method does not return the result of the last SYNCHRONIZE SQL statement. To obtain the SyncResult object for the last SYNCHRONIZE SQL statement, use the getSyncResult method on the Connection object passed in.

Related Information

SyncResult Class [page 224]
getSyncResult() Method [page 49]

1.25.13 getTableOrder() Method

Returns the order in which tables should be uploaded to the consolidated database.

Syntax

```
public abstract String getTableOrder ()
```

Returns

A comma separated list of table names; otherwise, returns null if a table order was not specified. See the class description for more information about comma separated lists.

Related Information

setTableOrder(String) Method [page 221]

1.25.14 getUserName() Method

Returns the MobiLink user name that uniquely identifies the client to the MobiLink server.

Syntax

```
public abstract String getUserName ()
```

Returns

The MobiLink user name.

Related Information

setUserName(String) Method [page 222]

1.25.15 getVersion() Method

Returns the script version to use.

Syntax

```
public abstract String getVersion ()
```

Returns

The script version.

Related Information

setVersion(String) Method [page 223]

1.25.16 isDownloadOnly() Method

Determines if the synchronization is download-only.

Syntax

```
public abstract boolean isDownloadOnly ()
```

Returns

True if uploads are disabled; otherwise, returns false.

Related Information

setDownloadOnly(boolean) Method [page 214]

1.25.17 isPingOnly() Method

Determines whether the client pings the MobiLink server or performs a synchronization.

Syntax

```
public abstract boolean isPingOnly ()
```

Returns

True if the client only pings the server; otherwise, returns false.

Related Information

setPingOnly(boolean) Method [page 218]

1.25.18 isUploadOnly() Method

Determines if the synchronization is upload-only.

Syntax

```
public abstract boolean isUploadOnly ()
```

Returns

True if downloads are disabled; otherwise, returns false.

Related Information

setUploadOnly(boolean) Method [page 221]

1.25.19 setAcknowledgeDownload(boolean) Method

Specifies whether the client should send download acknowledgements.

Syntax

```
public abstract void setAcknowledgeDownload (boolean ack)
```

Parameters

ack Set to true to have the client acknowledge a download; otherwise, set to false.

Remarks

The default is false.

Related Information

getAcknowledgeDownload() Method [page 202]

1.25.20 setAdditionalParms(String) Method

Specifies additional synchronization parameters as a semicolon-separated list of name=value pairs.

Syntax

```
public abstract void setAdditionalParms (String v) throws ULjException
```

Parameters

v A string, in the form of a semicolon-separated list of name=value pairs.

Remarks

Use this method to specify several additional synchronization parameters that cannot be specified using existing methods of the SyncParms class.

The following example illustrates how to set the AllowDownloadDupRows, CheckpointStore, and DisableConcurrency parameters on a SyncParms object:

```
SyncParms parms;
...
parms.setAdditionalParms(
  "AllowDownloadDupRows=1;CheckpointStore=1;DisableConcurrency=1" );
```

Related Information

getAdditionalParms() Method [page 202]

1.25.21 setAuthenticationParms(String) Method

Specifies parameters for a custom user authentication script (MobiLink authenticate_parameters connection event).

Syntax

```
public abstract void setAuthenticationParms (String v) throws ULjException
```

Parameters

v A comma separated list of authentication parameters, or the null reference. See the class description for more information about comma separated lists.

Remarks

Only the first 255 strings are used and each string should be no longer than the MobiLink server's limit for authentication parameters. (currently 4000 UTF8 bytes)

Strings longer than 21K characters are truncated when sent to MobiLink, and strings that exceed the server's limit for authentication parameters cause a server-side synchronization error.

Related Information

getAuthenticationParms() Method [page 203]

1.25.22 setDownloadOnly(boolean) Method

Sets the synchronization as download-only.

Syntax

```
public abstract void setDownloadOnly (boolean v)
```

Parameters

v Set to true to disable uploads, or set false to enable uploads.

Remarks

The default is false. Specifying true automatically calls the setPingOnly and setUploadOnly methods, and sets them to false.

Related Information

isDownloadOnly() Method [page 210]
setPingOnly(boolean) Method [page 218]
setUploadOnly(boolean) Method [page 221]

1.25.23 setKeepPartialDownload(boolean) Method

Specifies whether partial downloads should be allowed while synchronizing.

Syntax

```
public abstract void setKeepPartialDownload (boolean c) throws ULjException
```

Parameters

c Set to true to enable partial downloads.

Remarks

The default setting is false. Set to true to enable and save partial downloads while synchronizing; otherwise, set to false to disable partial downloads and roll back downloads if any errors occur.

UltraLite has the ability to resume partial downloads that fail due to communication errors or when the user aborts through the SyncObserver object. UltraLite processes the download as it is received. If a download is interrupted, then the partial download transaction remains in the database and can be resumed during the next synchronization.

To indicate that UltraLite should save partial downloads, set to true; otherwise, the download is rolled back if an error occurs.

If a partial download was kept, then the SyncResult.getPartialDownloadRetained method returns true when the Connection.synchronize method exits.

If the KeepPartialDownload synchronization parameter is set to true, then you can resume a partial download. To resume a partial download, call the Connection.synchronize method with the setResumePartialDownload method set to true.

Keep the KeepPartialDownload synchronization parameter set to true in case another communications error occurs. No upload is performed if a download is skipped.

The download you receive during a resumed download is as old as when the download originally began. If you need the most up to date data, then you can do another download immediately after the resumed download completes.

When resuming a download, many of the synchronization parameters that are specified by the SyncParms class are not relevant. For example, the Publications parameter is not used. You receive requested publications during the initial download. Only the setResumePartialDownload and setUserName methods need to be used. The setKeepPartialDownload method can be used if desired.

If you have a partial download and it is no longer needed, then you can call the Connection.rollbackPartialDownload to roll back the failed download transaction. Also, if you attempt to synchronize again and do not specify the ResumePartialDownload parameter, then the partial download is rolled back before the next synchronization begins.

Related Information

getKeepPartialDownload() Method [page 203]
setResumePartialDownload(boolean) Method [page 219]
setUserName(String) Method [page 222]

1.25.24 setLivenessTimeout(int) Method

Sets the liveness timeout length, in seconds.

Syntax

```
public abstract void setLivenessTimeout (int seconds) throws ULjException
```

Parameters

seconds The new liveness timeout value.

Remarks

The liveness timeout is the length of time the server allows a remote to be idle. If the remote does not communicate with the server for l seconds, the server assumes that the remote has lost the connection, and terminates the sync. The remote automatically sends periodic messages to the server to keep the connection alive.

If a negative value is set, an exception is thrown. The value may be changed by the MobiLink server without notice. This change occurs if the value is set too low or too high.

The default value is 240 seconds.

Related Information

getLivenessTimeout() Method [page 204]

1.25.25 setNewPassword(String) Method

Sets a new MobiLink password for the user specified with setUserName method.

Syntax

```
public abstract void setNewPassword (String v)
```

Parameters

v A new password for MobiLink user.

Remarks

The new password takes effect after the next synchronization.

The default is null, suggesting that the password does not get replaced.

Related Information

getNewPassword() Method [page 204]
setPassword(String) Method [page 217]
setUserName(String) Method [page 222]

1.25.26 setPassword(String) Method

Sets the MobiLink password for the user specified with the setUserName method.

Syntax

```
public abstract void setPassword (String v) throws ULjException
```

Parameters

v A password for the MobiLink user.

Remarks

This user name and password is separate from any database user ID and password. This method is used to authenticate the application against the MobiLink server.

The default is an empty string, suggesting no password.

Related Information

getPassword() Method [page 205]
setNewPassword(String) Method [page 216]
setUserName(String) Method [page 222]

1.25.27 setPingOnly(boolean) Method

Sets the client to ping the MobiLink server rather than perform a synchronization.

Syntax

```
public abstract void setPingOnly (boolean v)
```

Parameters

v Set to true to only ping the server, or set false to perform a synchronization.

Remarks

The default is false. Specifying true automatically calls the setDownloadOnly and setUploadOnly methods, and sets them to false.

Related Information

isPingOnly() Method [page 211]
setDownloadOnly(boolean) Method [page 214]
setUploadOnly(boolean) Method [page 221]

1.25.28 setPublications(String) Method

Sets the publications to be synchronized.

Syntax

```
public abstract void setPublications (String pubs) throws ULjException
```

Parameters

pubs A comma separated list of publication names. See the class description for more information about comma separated lists.

Remarks

The default is set to the Connection.SYNC_ALL constant, which is used to denote the synchronization of all tables in the database. To synchronize all publications, set this method to the Connection.SYNC_ALL_PUBS constant.

Related Information

getPublications() Method [page 205]

1.25.29 setResumePartialDownload(boolean) Method

Specifies whether to resume or discard a previous partial download.

Syntax

```
public abstract void setResumePartialDownload (boolean c) throws ULjException
```

Parameters

c Set to true to resume a previous partial download.

Exceptions

ULjException class SQLE_SYNC_INFO_INVALID is thrown by the Connection.synchronize method when more than one of the following synchronization parameters (DownloadOnly, PingOnly, ResumePartialDownload, or UploadOnly) is set to true.

Remarks

Set to true to resume a previous partial download, or false to discard a previous partial download. The default setting is false.

Related Information

getResumePartialDownload() Method [page 206]

1.25.30 setSyncObserver(SyncObserver) Method

Sets a SyncObserver object to monitor the progress of the synchronization.

Syntax

```
public abstract void setSyncObserver (SyncObserver so)
```

Parameters

so A SyncObserver object.

Remarks

The default is null, suggesting no observer.

Related Information

SyncObserver Interface [page 193]

1.25.31 setTableOrder(String) Method

Sets the order in which tables should be uploaded to the consolidated database.

Syntax

```
public abstract void setTableOrder (String v) throws ULjException
```

Parameters

v A comma separated list of table names in the order they should be synchronized, or null, indicating no table order. See the class description for more information about comma separated lists.

Remarks

The primary table should be listed first, along with all tables containing foreign key relationships in the consolidated database.

All tables selected for synchronization by the Publications parameter are synchronized whether they are specified in the TableOrder parameter or not. Unspecified tables are synchronized by order of the foreign key relations in the client database. They are synchronized after the specified tables.

The default is a null reference, which does not override the default ordering of tables.

Related Information

getTableOrder() Method [page 209]
setPublications(String) Method [page 218]

1.25.32 setUploadOnly(boolean) Method

Sets the synchronization as upload-only.

Syntax

```
public abstract void setUploadOnly (boolean v)
```

Parameters

v Set to true to disable downloads, or set or false to enable downloads.

Remarks

The default is false. Specifying true automatically calls the setDownloadOnly and setPingOnly methods, and sets them to false.

Related Information

isUploadOnly() Method [page 211]
setDownloadOnly(boolean) Method [page 214]
setPingOnly(boolean) Method [page 218]

1.25.33 setUserName(String) Method

Sets the MobiLink user name that uniquely identifies the client to the MobiLink server.

Syntax

```
public abstract void setUserName (String v) throws ULjException
```

Parameters

v The MobiLink user name.

Remarks

This value is used to determine the following:

- Download content
- Whether to record the synchronization state
- Whether to recover from interruptions during synchronization.

This user name and password is separate from any database user ID and password. This method is used to authenticate the application against the MobiLink server.

This parameter is initialized when the SyncParms object is created.

Related Information

getUserName() Method [page 209]
setPassword(String) Method [page 217]
setNewPassword(String) Method [page 216]
createSyncParms(int, String, String) Method [page 43]

1.25.34 setVersion(String) Method

Sets the synchronization script to use.

Syntax

```
public abstract void setVersion (String v) throws ULjException
```

Parameters

v The script version.

Remarks

Each synchronization script in the consolidated database is marked with a version string. For example, there can be two different download_cursor scripts, and each one is identified by different version strings. The version string allows an application to choose from a set of synchronization scripts.

This parameter is initialized when the SyncParms object is created.

Related Information

getVersion() Method [page 210]
createSyncParms(int, String, String) Method [page 43]

1.26 SyncResult Class

Reports status-related information about a specified database synchronization.

Syntax

```
public abstract class SyncResult
```

Members

All members of SyncResult, including inherited members.

Methods

Modifier and Type	Method	Description
public abstract String	getAuthMessage() [page 227]	Returns the authorization message of the last synchronization attempt as specified by custom user authentication synchronization scripts.
public abstract int	getAuthStatus() [page 227]	Returns the authorization status code of the last synchronization attempt.
public abstract int	getAuthValue() [page 228]	Returns the value specified in custom user authentication synchronization scripts.
public abstract String	getCurrentTableName() [page 228]	Returns the name of the table currently being synchronized.
public abstract boolean	getIgnoredRows() [page 228]	Determines if any uploaded rows were ignored during the last synchronization.
public abstract boolean	getPartialDownloadRetained() [page 229]	Checks whether a partial download was retained during the last synchronization.
public abstract long	getReceivedByteCount() [page 229]	Returns the number of bytes received during data synchronization.
public abstract long	getReceivedDeletes() [page 229]	Returns the number of received rows that have been deleted.
public abstract long	getReceivedIgnoredDeletes() [page 230]	Returns the number of received delete rows that have been ignored.
public abstract long	getReceivedIgnoredUpdates() [page 230]	Returns the number of received update rows that have been ignored.
public abstract long	getReceivedInserts() [page 231]	Returns the number of received rows that have been inserted.
public abstract long	getReceivedRowCount() [page 232]	Returns the number of rows received.

Modifier and Type	Method	Description
public abstract long	getReceivedTruncateDeletes() [page 232]	Returns the number of rows that have been deleted by a downloaded truncate operation.
public abstract long	getReceivedUpdates() [page 233]	Returns the number of received rows that have been applied as updates.
public abstract long	getSentByteCount() [page 234]	Returns the number of bytes sent during data synchronization.
public abstract long	getSentDeletes() [page 234]	Returns the number of deleted rows sent.
public abstract long	getSentInserts() [page 235]	Returns the number of inserted rows sent.
public abstract long	getSentUpdates() [page 235]	Returns the number of updated rows sent.
public abstract int	getStreamErrorCode() [page 236]	Returns the error code reported by the stream itself.
public abstract String	getStreamErrorMessage() [page 236]	Returns the error message reported by the stream itself.
public abstract int	getSyncedTableCount() [page 237]	Returns the number of synchronized tables so far.
public abstract long	getTotalDownloadRowCount() [page 237]	Returns the total number of rows to be received in the download.
public abstract int	getTotalTableCount() [page 238]	Returns the number of tables to be synchronized.
public abstract boolean	isUploadOK() [page 238]	Determines if the last upload synchronization was successful.

In this section:

Returns the number of bytes received during data synchronization.

getReceivedDeletes() Method [page 229]
: Returns the number of received rows that have been deleted.

getReceivedIgnoredDeletes() Method [page 230]
: Returns the number of received delete rows that have been ignored.

getReceivedIgnoredUpdates() Method [page 230]
: Returns the number of received update rows that have been ignored.

getReceivedInserts() Method [page 231]
: Returns the number of received rows that have been inserted.

getReceivedRowCount() Method [page 232]
: Returns the number of rows received.

getReceivedTruncateDeletes() Method [page 232]
: Returns the number of rows that have been deleted by a downloaded truncate operation.

getReceivedUpdates() Method [page 233]
: Returns the number of received rows that have been applied as updates.

getSentByteCount() Method [page 234]
: Returns the number of bytes sent during data synchronization.

getSentDeletes() Method [page 234]
: Returns the number of deleted rows sent.

getSentInserts() Method [page 235]
: Returns the number of inserted rows sent.

getSentUpdates() Method [page 235]
: Returns the number of updated rows sent.

getStreamErrorCode() Method [page 236]
: Returns the error code reported by the stream itself.

getStreamErrorMessage() Method [page 236]
: Returns the error message reported by the stream itself.

getSyncedTableCount() Method [page 237]
: Returns the number of synchronized tables so far.

getTotalDownloadRowCount() Method [page 237]
: Returns the total number of rows to be received in the download.

getTotalTableCount() Method [page 238]
: Returns the number of tables to be synchronized.

isUploadOK() Method [page 238]
: Determines if the last upload synchronization was successful.

Related Information

getSyncResult() Method [page 208]

1.26.1 getAuthMessage() Method

Returns the authorization message of the last synchronization attempt as specified by custom user authentication synchronization scripts.

Syntax

```
public abstract String getAuthMessage ()
```

Returns

A string containing information regarding the authentication of the last synchronization.

Remarks

Blank or empty messages are returned as null.

An authentication message may be returned for any authorization status code. See the authentication_message MobiLink named system parameter for more detail.

1.26.2 getAuthStatus() Method

Returns the authorization status code of the last synchronization attempt.

Syntax

```
public abstract int getAuthStatus ()
```

Returns

An AuthStatusCode value.

1.26.3 getAuthValue() Method

Returns the value specified in custom user authentication synchronization scripts.

Syntax

```
public abstract int getAuthValue ()
```

Returns

An integer returned from custom user authentication synchronization scripts.

1.26.4 getCurrentTableName() Method

Returns the name of the table currently being synchronized.

Syntax

```
public abstract String getCurrentTableName ()
```

Returns

The table name.

1.26.5 getIgnoredRows() Method

Determines if any uploaded rows were ignored during the last synchronization.

Syntax

```
public abstract boolean getIgnoredRows ()
```

Returns

True if any uploaded rows were ignored during the last synchronization; otherwise, returns false if no rows were ignored.

1.26.6 getPartialDownloadRetained() Method

Checks whether a partial download was retained during the last synchronization.

Syntax

```
public abstract boolean getPartialDownloadRetained ()
```

Returns

True if a download was interrupted and the partial download was retained, or false if the download was not interrupted or if the partial download was rolled back.

1.26.7 getReceivedByteCount() Method

Returns the number of bytes received during data synchronization.

Syntax

```
public abstract long getReceivedByteCount ()
```

Returns

The number of bytes.

1.26.8 getReceivedDeletes() Method

Returns the number of received rows that have been deleted.

Syntax

```
public abstract long getReceivedDeletes ()
```

Returns

The number of downloaded rows applied as deletes.

Related Information

getReceivedIgnoredDeletes() Method [page 230]
getReceivedIgnoredUpdates() Method [page 230]
getReceivedInserts() Method [page 231]
getReceivedTruncateDeletes() Method [page 232]
getReceivedUpdates() Method [page 233]

1.26.9 getReceivedIgnoredDeletes() Method

Returns the number of received delete rows that have been ignored.

Syntax

```
public abstract long getReceivedIgnoredDeletes ()
```

Returns

The number of downloaded delete rows that were ignored.

Related Information

getReceivedIgnoredUpdates() Method [page 230]
getReceivedDeletes() Method [page 229]
getReceivedInserts() Method [page 231]
getReceivedTruncateDeletes() Method [page 232]
getReceivedUpdates() Method [page 233]

1.26.10 getReceivedIgnoredUpdates() Method

Returns the number of received update rows that have been ignored.

Syntax

```
public abstract long getReceivedIgnoredUpdates ()
```

Returns

The number of downloaded update rows that were ignored.

Remarks

Received update rows are ignored only when duplicate primary keys are allowed in the download. Duplicate downloaded updates otherwise result in a failed synchronization.

Related Information

getReceivedIgnoredDeletes() Method [page 230]
getReceivedDeletes() Method [page 229]
getReceivedInserts() Method [page 231]
getReceivedTruncateDeletes() Method [page 232]
getReceivedUpdates() Method [page 233]

1.26.11 getReceivedInserts() Method

Returns the number of received rows that have been inserted.

Syntax

```
public abstract long getReceivedInserts ()
```

Returns

The number of rows applied as inserts.

Related Information

getReceivedDeletes() Method [page 229]
getReceivedIgnoredDeletes() Method [page 230]
getReceivedTruncateDeletes() Method [page 232]
getReceivedUpdates() Method [page 233]

1.26.12 getReceivedRowCount() Method

Returns the number of rows received.

Syntax

```
public abstract long getReceivedRowCount ()
```

Returns

The number of rows received.

Remarks

This count includes rows that may be ignored when the download is applied.

Related Information

getTotalDownloadRowCount() Method [page 237]

1.26.13 getReceivedTruncateDeletes() Method

Returns the number of rows that have been deleted by a downloaded truncate operation.

Syntax

```
public abstract long getReceivedTruncateDeletes ()
```

Returns

The number of rows truncated.

Remarks

Each downloaded truncate operation appears as a single row in the row download as counted by the getReceivedRowCount method but may result in zero or many rows being truncated, as counted by this method.

Related Information

getReceivedDeletes() Method [page 229]
getReceivedIgnoredDeletes() Method [page 230]
getReceivedIgnoredUpdates() Method [page 230]
getReceivedInserts() Method [page 231]
getReceivedUpdates() Method [page 233]

1.26.14 getReceivedUpdates() Method

Returns the number of received rows that have been applied as updates.

Syntax

```
public abstract long getReceivedUpdates ()
```

Returns

The number of rows applied as updates.

Related Information

getReceivedDeletes() Method [page 229]
getReceivedIgnoredDeletes() Method [page 230]
getReceivedIgnoredUpdates() Method [page 230]
getReceivedInserts() Method [page 231]
getReceivedTruncateDeletes() Method [page 232]

1.26.15 getSentByteCount() Method

Returns the number of bytes sent during data synchronization.

Syntax

```
public abstract long getSentByteCount ()
```

Returns

The number of bytes sent.

1.26.16 getSentDeletes() Method

Returns the number of deleted rows sent.

Syntax

```
public abstract long getSentDeletes ()
```

Returns

The number of deleted rows sent.

Remarks

The number of inserts, updates, and deletes may differ than the number of operations performed on the tables being synchronized because all operations on a given row are coalesced into one.

Related Information

getSentInserts() Method [page 235]
getSentUpdates() Method [page 235]

1.26.17 getSentInserts() Method

Returns the number of inserted rows sent.

Syntax

```
public abstract long getSentInserts ()
```

Returns

The number of inserted rows sent.

Remarks

The number of inserts, updates, and deletes may differ than the number of operations performed on the tables being synchronized because all operations on a given row are coalesced into one.

Related Information

getSentDeletes() Method [page 234]
getSentUpdates() Method [page 235]

1.26.18 getSentUpdates() Method

Returns the number of updated rows sent.

Syntax

```
public abstract long getSentUpdates ()
```

Returns

The number of updated rows sent.

Remarks

The number of inserts, updates, and deletes may differ than the number of operations performed on the tables being synchronized because all operations on a given row are coalesced into one.

Related Information

getSentDeletes() Method [page 234]
getSentInserts() Method [page 235]

1.26.19 getStreamErrorCode() Method

Returns the error code reported by the stream itself.

Syntax

```
public abstract int getStreamErrorCode ()
```

Returns

0 if there was no communication stream error; otherwise, returns the response code from the server.

Remarks

This method returns the HTTP response code.

1.26.20 getStreamErrorMessage() Method

Returns the error message reported by the stream itself.

Syntax

```
public abstract String getStreamErrorMessage ()
```

Returns

Null if no message is available; otherwise, returns the response message.

Remarks

This method returns the HTTP response message.

1.26.21 getSyncedTableCount() Method

Returns the number of synchronized tables so far.

Syntax

```
public abstract int getSyncedTableCount ()
```

Returns

The number of tables synchronized.

1.26.22 getTotalDownloadRowCount() Method

Returns the total number of rows to be received in the download.

Syntax

```
public abstract long getTotalDownloadRowCount ()
```

Returns

The number of rows to be received in the download. This number includes any rows that do not apply, such as deletes for rows that are not on the client.

Remarks

This number includes duplicate rows that are ignored. This value is not set until the synchronization enters the SyncObserver.State.RECEIVING_TABLE state for the first table.

1.26.23 getTotalTableCount() Method

Returns the number of tables to be synchronized.

Syntax

```
public abstract int getTotalTableCount ()
```

Returns

The number of tables to synchronize.

1.26.24 isUploadOK() Method

Determines if the last upload synchronization was successful.

Syntax

```
public abstract boolean isUploadOK ()
```

Returns

True if the last upload synchronization was successful; otherwise, returns false.

1.27 SyncResult.AuthStatusCode Interface

Enumerates the authorization codes returned by the MobiLink server.

Syntax

```
public interface AuthStatusCode
```

Members

All members of AuthStatusCode, including inherited members.

Variables

Modifier and Type	Variable	Description
public final int	UNKNOWN	Authorization status is unknown. This code suggests that a synchronization has not been performed.
public final int	VALID	User ID and password were valid at time of synchronization.
public final int	VALID_BUT_EXPIRES_SOON	User ID and password were valid at time of synchronization but expire soon.
public final int	EXPIRED	User ID or password has expired. Authorization fails.
public final int	INVALID	Bad user ID or password. Authorization fails.
public final int	IN_USE	User ID is already in use. Authorization fails.

Related Information

getAuthStatus() Method [page 227]

1.28 TableSchema Interface

Specifies the schema of a table and provides constants defining the names of system tables.

Syntax

```
public interface TableSchema
```

Members

All members of TableSchema, including inherited members.

Variables

Modifier and Type	Variable	Description
public static final String	SYS_TABLES	Contains the name of the system table containing information about the tables in the database.
public static final String	SYS_COLUMNS	Contains the name of the system table containing information about the table columns in the database.
public static final String	SYS_INDEXES	Contains the name of the system table containing information about the table indexes in the database.
public static final String	SYS_INDEX_COLUMNS	Contains the name of the system table containing information about the index columns in the database.
public static final String	SYS_PUBLICATIONS	Contains the name of the system table containing information about database publications.
public static final String	SYS_ARTICLES	Contains the name of the system table containing information about publica-tion articles.
public static final String	SYS_INTERNAL	Contains the name of the system table containing internal information.
public static final String	SYS_FOREIGN_KEYS	Contains the name of the system table containing information about foreign keys in the database.
public static final String	SYS_FKEY_COLUMNS	Contains the name of the system table containing information about foreign key columns.
public static final String	SYS_ULDATA	Contains the name of the system table containing information about system values.
public static final String	SYS_ULDATA_INTERNAL	Contains the type for internal system data.
public static final String	SYS_ULDATA_OPTION	Contains the type for option system data.
public static final String	SYS_ULDATA_PROPERTY	Contains the type for property system data.
public static final String	SYS_PRIMARY_INDEX	Contains the name of the primary key index of system tables.
public static final short	TABLE_IS_SYSTEM	Denotes that a table is a system table. This value can be logically combined with other flags in the table_flags col-umn of the SYS_TABLES table.

Modifier and Type	Variable	Description
public static final short	TABLE_IS_NOSYNC	Denotes that a table is a non-synchronizing table. This value can be logically combined with other flags in the table_flags column of the SYS_TABLES table except the TABLE_IS_DOWNLOAD_ONLY flag.
public static final short	TABLE_IS_DOWNLOAD_ONLY	Denotes that a table is a download-only table (a table that is uploaded when synchronized). This value can be logically combined with other flags in the table_flags column of the SYS_TABLES table except the TABLE_IS_NOSYNC flag.

Remarks

This interface only contains table-related constants. They include system table names, table flags, and types of data in the *sysuldata* system table.

1.29 ULjEvent Interface

Represents an UltraLiteJ API system event.

Syntax

```
public interface ULjEvent
```

Members

All members of ULjEvent, including inherited members.

Variables

Modifier and Type	Variable	Description
public final short	TABLE_MODIFIED_EVENT	Indicates the "Table modified" event type.
public final short	COMMIT_EVENT	Indicates the "Commit" event type.

Modifier and Type	Variable	Description
public final short	SYNC_COMPLETE_EVENT	Indicates the "Synchronization complete" event type.

Methods

Modifier and Type	Method	Description
public String	getParameter(String) [page 242]	Returns a named parameter for the event.
public short	getType() [page 242]	Returns the event type.

In this section:

getParameter(String) Method [page 242]
    Returns a named parameter for the event.

getType() Method [page 242]
    Returns the event type.

1.29.1 getParameter(String) Method

Returns a named parameter for the event.

Syntax

```
public String getParameter (String name) throws ULjException
```

Parameters

name The event name to get the value for.

1.29.2 getType() Method

Returns the event type.

Syntax

```
public short getType ()
```

1.30 ULjException Class

Supersedes the exceptions thrown by the database.

Syntax

```
public abstract class ULjException
```

Members

All members of ULjException, including inherited members.

Constructors

Modifier and Type	Constructor	Description
protected	ULjException(String) [page 244]	Constructs a new ULjException class.

Methods

Modifier and Type	Method	Description
public abstract ULjException	getCausingException() [page 244]	Returns the ULjException object when an exception is caused.
public abstract int	getErrorCode() [page 244]	Returns the error code associated with the exception.
public abstract String	getParameter(short) [page 245]	Returns the specified error parameter.
public abstract short	getParameterCount() [page 245]	Returns the number of error parameters.
public abstract int	getSqlOffset() [page 246]	Returns the error offset within the SQL string.

In this section:

Returns the error offset within the SQL string.

1.30.1 ULjException(String) Constructor

Constructs a new ULjException class.

Syntax

```
protected ULjException (String text)
```

Parameters

text The message text.

1.30.2 getCausingException() Method

Returns the ULjException object when an exception is caused.

Syntax

```
public abstract ULjException getCausingException ()
```

Returns

Null if no causing exceptions exist; otherwise, returns the ULjException object.

1.30.3 getErrorCode() Method

Returns the error code associated with the exception.

Syntax

```
public abstract int getErrorCode ()
```

Returns

The error code.

1.30.4 getParameter(short) Method

Returns the specified error parameter.

Syntax

```
public abstract String getParameter (short param_no)
```

Parameters

param_no A one-based parameter number.

Returns

The error parameter.

1.30.5 getParameterCount() Method

Returns the number of error parameters.

Syntax

```
public abstract short getParameterCount ()
```

Returns

The number of error parameters.

1.30.6 getSqlOffset() Method

Returns the error offset within the SQL string.

Syntax

```
public abstract int getSqlOffset ()
```

Returns

-1 when there is no SQL string associated with the error message; otherwise, returns the zero-base offset within that string where the error occurred.

1.31 Unsigned64 Class

Implements unsigned 64-bit binary values.

Syntax

```
public class Unsigned64
```

Members

All members of Unsigned64, including inherited members.

Methods

Modifier and Type	Method	Description
public static final long	add(long, long) [page 247]	Adds two values together and places the result in self.
public static final byte	compare [page 248]	Compares two integer values.
public static final long	divide(long, long) [page 249]	Divides two values and places the result in self.
public static final long	multiply(long, long) [page 250]	Multiplies two values together and places the result in self.
public static final long	remainder [page 251]	Returns the remainder when one value is divided by another.

Modifier and Type	Method	Description
public static final long	subtract(long, long) [page 252]	Subtracts two values and places the result in self.

Remarks

The intent of this class is to keep values as long integers and interpret them using the static methods in this class.

This class cannot be instantiated.

In this section:

1.31.1 add(long, long) Method

Adds two values together and places the result in self.

Syntax

```
public static final long add (
    long v1,
    long v2
)
```

Parameters

v1 The first operand

v2 The second operand

Returns

The sum of the operands.

1.31.2 compare Method

Compares two integer values.

Overload list

Modifier and Type	Overload name	Description
public static final byte	compare(int, int) [page 248]	Compares two integer values.
public static final byte	compare(long, long) [page 249]	Compares two long values.

In this section:

compare(int, int) Method [page 248]
    Compares two integer values.

compare(long, long) Method [page 249]
    Compares two long values.

1.31.2.1 compare(int, int) Method

Compares two integer values.

Syntax

```
public static final byte compare (
     int v1,
     int v2
)
```

Parameters

v1 The first value to be compared.

v2 The second value to be compared.

Returns

-1 when v2 is greater than v1, 0 when v1 equals v2, or 1 when v2 is less than v1.

1.31.2.2 compare(long, long) Method

Compares two long values.

Syntax

```
public static final byte compare (
    long v1,
    long v2
)
```

Parameters

v1 The first value to be compared.
v2 The second value to be compared.

Returns

-1 when v2 is greater than v1, 0 when v1 equals v2, or 1 when v2 is less than v1.

1.31.3 divide(long, long) Method

Divides two values and places the result in self.

Syntax

```
public static final long divide (
    long v1,
    long v2
)
```

Parameters

v1 The first operand

v2 The second operand

Returns

The first operand divided by the second operand.

1.31.4 multiply(long, long) Method

Multiplies two values together and places the result in self.

Syntax

```
public static final long multiply (
    long v1,
    long v2
)
```

Parameters

v1 The first operand.
v2 The second operand.

Returns

The product of v1 and v2.

1.31.5 remainder Method

Returns the remainder when one value is divided by another.

Overload list

Modifier and Type	Overload name	Description
public static final long	remainder(long, long) [page 251]	Returns the remainder when one value is divided by another.
public static final long	remainder(long, long, long) [page 252]	Returns the remainder when one value multiplied by a given quotient is subtracted from another value (v1 - quot * v2).

In this section:

remainder(long, long) Method [page 251]
: Returns the remainder when one value is divided by another.

remainder(long, long, long) Method [page 252]
: Returns the remainder when one value multiplied by a given quotient is subtracted from another value (v1 - quot * v2).

1.31.5.1 remainder(long, long) Method

Returns the remainder when one value is divided by another.

Syntax

```
public static final long remainder (
    long v1,
    long v2
)
```

Parameters

v1 The value to be divided.

v2 The value to divide by.

Returns

The remainder, represented as a long integer.

1.31.5.2 remainder(long, long, long) Method

Returns the remainder when one value multiplied by a given quotient is subtracted from another value (v1 - quot * v2).

Syntax

```
public static final long remainder (
    long v1,
    long v2,
    long quot
)
```

Parameters

v1 The value to be divided.
v2 The value to divide by.
quot The value of the quotient.

Returns

The remainder, represented as a long integer.

1.31.6 subtract(long, long) Method

Subtracts two values and places the result in self.

Syntax

```
public static final long subtract (
    long v1,
    long v2
)
```

Parameters

v1 The first operand.
v2 The second operand.

Returns

The result of v2 being subtracted from v1.

1.32 UUIDValue Interface

Describes a unique identifier (UUID or Universally Unique IDentifier) object.

Syntax

```
public interface UUIDValue
```

Members

All members of UUIDValue, including inherited members.

Methods

Modifier and Type	Method	Description
public String	getString() [page 254]	Returns the String representation of the UUIDValue object.
public boolean	isNull() [page 254]	Determines if the UUIDValue object is null.
public void	set(String) [page 255]	Sets the UUIDValue object with a String value.
public void	setNull() [page 255]	Sets the UUIDValue object to null.

Remarks

Such entities are useful when a unique identifier is required and where the value can be arbitrary.

A UUIDValue can also be created by the SQL INSERT statement when no value is supplied for a column in a table where the column was created with the DEFAULT NEWID() clause.

A Connection object can be used to create a UUIDValue using the createUUIDValue method.

In this section:

Related Information

createUUIDValue() Method [page 44]

1.32.1 getString() Method

Returns the String representation of the UUIDValue object.

Syntax

```
public String getString () throws ULjException
```

Returns

The String value.

1.32.2 isNull() Method

Determines if the UUIDValue object is null.

Syntax

```
public boolean isNull ()
```

Returns

True if the object is null; otherwise, returns false.

1.32.3 set(String) Method

Sets the UUIDValue object with a String value.

Syntax

```
public void set (String value) throws ULjException
```

Parameters

value A numerical value represented as a String.

1.32.4 setNull() Method

Sets the UUIDValue object to null.

Syntax

```
public void setNull () throws ULjException
```

1.33 ValidateDatabaseProgressData Interface

Reports ValidateDatabase progress data.

Syntax

```
public interface ValidateDatabaseProgressData
```

Members

All members of ValidateDatabaseProgressData, including inherited members.

Methods

Modifier and Type	Method	Description
public abstract String[]	getParms() [page 256]	Returns a parameter array associated with the status ID.
public abstract short	getStatusId() [page 256]	Returns the status ID of the validation operation.

In this section:

getParms() Method [page 256]
Returns a parameter array associated with the status ID.

getStatusId() Method [page 256]
Returns the status ID of the validation operation.

1.33.1 getParms() Method

Returns a parameter array associated with the status ID.

Syntax

```
public abstract String[] getParms ()
```

Returns

The array of parameters as a fixed size; unused parameters are null.

Related Information

ValidateDatabaseProgressData.StatusId Interface [page 257]

1.33.2 getStatusId() Method

Returns the status ID of the validation operation.

Syntax

```
public abstract short getStatusId ()
```

Returns

A status ID constant.

1.34 ValidateDatabaseProgressData.StatusId Interface

Specifies possible status IDs for the UltraLite Validate Database utility.

Syntax

```
public interface StatusId
```

Members

All members of StatusId, including inherited members.

Variables

Modifier and Type	Variable	Description
public final short	UL_VALID_NO_ERROR	No error occurred.
public final short	UL_VALID_START	Start validation.
public final short	UL_VALID_END	End validation. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks the resulting SQLCODE, which indicates success or failure.
public final short	UL_VALID_CHECKING_PAGE	Send a periodic status message while checking database pages. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks a number associated with the page. The order is not defined.
public final short	UL_VALID_CHECKING_TABLE	Checking a table. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks the table name.

Modifier and Type	Variable	Description
public final short	UL_VALID_CHECKING_INDEX	Checking an index. The first parameter returned by the ValidateDatabaseProgressData.getParms method stores the table name. The second parameter stores the index name.
public final short	UL_VALID_DATABASE_ERROR	An error occurred accessing the database. Check the SQLCODE for more information.
public final short	UL_VALID_STARTUP_ERROR	Error starting the database for low-level access.
public final short	UL_VALID_CONNECT_ERROR	Error connecting to the database.
public final short	UL_VALID_INTERRUPTED	Validation process interrupted.
public final short	UL_VALID_CORRUPT_PAGE_TABLE	Page table is corrupt.
public final short	UL_VALID_FAILED_CHECKSUM	Page checksum failed. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks a number associated with the page.
public final short	UL_VALID_CORRUPT_PAGE	A page is corrupt. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks a number associated with the page.
public final short	UL_VALID_ROWCOUNT_MISMATCH	The number of rows in the index is different from the table row count. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks the table name. The second parameter tracks index name.
public final short	UL_VALID_BAD_ROWID	There is an invalid row identifier in the index. The first parameter returned by the ValidateDatabaseProgressData.getParms method tracks the table name. The second parameter tracks the index name.

1.35 ValidateDatabaseProgressListener Interface

Receives ValidateDatabase progress events.

Syntax

```
public interface ValidateDatabaseProgressListener
```

Members

All members of ValidateDatabaseProgressListener, including inherited members.

Methods

Modifier and Type	Method	Description
public boolean	validateProgressed(ValidateDatabase-ProgressData) [page 259]	Gets invoked during a ValidateDatabase operation to inform the user of the validation progress.

In this section:

validateProgressed(ValidateDatabaseProgressData) Method [page 259]
Gets invoked during a ValidateDatabase operation to inform the user of the validation progress.

1.35.1 validateProgressed(ValidateDatabaseProgressData) Method

Gets invoked during a ValidateDatabase operation to inform the user of the validation progress.

Syntax

```
public boolean validateProgressed (ValidateDatabaseProgressData data)
```

Parameters

data A ValidateDatabaseProgressData object containing the latest validate progress data.

Returns

True to cancel the validation process; otherwise, returns false.

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.

www.sap.com/contactsap

THE BEST RUN