

---

## SimbaEngine X 10.1.11

Released 2018-11-02

These release notes provide details of features and known issues in SimbaEngine SDK X version 10.1.11.

### Enhancements & New Features

#### Support for parsing optional scalar paramaters in Java SQL Engine

[SEN SQLEngine] With this release, drivers support parsing precision and scale parameters for the scalar methods CAST and CONVERT.

#### Improved logging

[Client] JDBC client now supports logging. Various other minor enhancements have been made to logging.

#### Support for binary literals in Java SQL Engine

[SQLEngine][JSQLEngine]: Binary literals are now supported in Java SQL Engine queries.

#### Java SQL engine data type conversion

[Support][JSQLEngine] The Java SQL engine now supports conversion from VARCHAR to BINARY.

#### Improved C/SQL type support

[ODBC] A `SqlBindParameterPreHook()` method has been added to `ODBCSemantics` to allow changing C/SQL type after binding.

#### Enhanced support for Turkish locales

[Support][DSI][ODBC] To improve support for Turkish locales, overloads have been added to the comparison functions of `simba_wstring`. This allows you to specify what behavior to use when doing case-insensitive comparisons with

```
simba_wstring::CaseInsensitiveBehaviour.
```

There are now two modes:

- The original behavior (CIB\_DEFAULT)
- The new behaviour (CIB\_EQUATE\_DOTTED\_AND\_DOTLESS\_I), where dotted & dotless I's (i or ı) are considered to be equivalent (regardless of case).

A state has been added to `simba_wstring::CaseInsensitiveComparator` to allow you to set the behavior using `simba_wstring::CaseInsensitiveBehaviour`. CIB\_DEFAULT is the default behavior.

A locale can now be specified when calling `simba_wstring::ToUpper/`. For now, you can choose between the default locale (the current behavior), and explicitly specifying the US locale.

You should use `U_FOLD_CASE_EXCLUDE_SPECIAL_I` when calling `UnicodeString::caseCompare()` in Turkish locales so that `simba_wstring::ToUpper/ToLower` are consistent with `simba_wstring`'s case-insensitive compares.

The method

`ODBCSemantics::GetCaseInsensitiveBehaviourForConnSettings()` has been added to allow you to control which behavior (as described above) is used for connection setting request & response maps when attempting an ODBC connection.

A new function, `SimbaSettingReader::OverrideKeyComparator`, allows the DSII to customize how `SimbaSettingReader` compares keys.

`IniFileConfigurationReader` is updated to use the comparator of the map passed into it.

### New methods

- [CLIDSI] A `Simba::CLIDSI::SetODBCSemanticsFactory()` method has been added to allow CLIDSI-based drivers to override `Simba::DSI::IDriver::CreateODBCSemantics()`.
- [DSI][ODBC] A new `ODBCSemantics::SqlBindParameterPostHook()` method allows the DSII to modify the binding to work around application errors.

### Improved transaction workflow support

[DSI][ODBC] A new `ODBCSemantics::AllowFakeTransactions()` method has been added. This allows support for applications which unconditionally disable autocommit without checking whether the driver supports transactions.

This is accomplished by the DSI telling the ODBC layer to pretend to allow auto-commit to be disabled. This only works if the application does not attempt to rollback transactions, or make use of some other product of transactions: isolation for example.

### Improved parameter count support

[CLIDS] Add support for non-zero `DSI_CONN_ODBC_VALIDATE_PARAMETER_COUNT` for drivers that support SQL capable data sources.

### Improved hybrid hash support

[SEN] Drivers can handle arithmetic expressions in a hybrid hash.

### Enhanced JDBC read/write support

[JDBC] Support has been added for `java.sql.Types.STRUCT` and `java.sql.Types.JAVA_OBJECT` types in JDBC, enabling drivers to read and write Structs and objects (such as Maps) to and from databases.

## Changes and Resolved Issues

[Support] A `getLocaleLanguage()` method has been added to allow drivers to determine the current process language.

[SEN] In some cases, the server would not shut down properly and stop responding.

[SEN] Drivers attempt to call the deprecated `TypeConversionInfo::GetInstance()` function, which causes warnings to display.

[ClientServer] Data compression issues can occur when data is sent from the server to the client.

[JSQLEngine] Metadata issues can occur when using the `Convert()` function to convert from CHAR to BINARY.

[DSI] Incorrect source encoding is used when converting a string to the target encoding used in a `SqlData` object.

[SEN-5840] Column names may not be correct after passthrough handling when a correlation specification or derived column list are used in the query.

[ODBC] Error messages are not posted to the warning listener when the implementation of `SQLEndTran` throws an exception.

[Server] In some cases, an assertion failure occurs when the server is shutting down.

[Server] Requests are stopped when cancelling them. This prevents the server waiting on them for a long time while trying to shut down or close a connection.

[CLIDSII] In some cases, CLIDSII on drivers that do not use SQLEngine fails to marshall default parameter values to the managed DSII.

[DotNetDSII] `Simba.DotNetDSII.PropertyValues.DSI_OVPC_ALLOW_FEWER` and `Simba.DotNetDSII.PropertyValues.DSI_OVPC_ALLOW_MORE` are missing constants.

[SEN-5812][00132470][JavaDSII] In some cases, the `DSIMetadataResultSet.compareTo()` method can encounter an unexpected `NullPointerException` when working with `BIGINTEGER` data types.

[DSII][DSIExt][CLIDSII] Corrected possible use-after-free vulnerabilities for CLIDSII-based drivers.

[SEN SQLEngine] The refactored `DSIExtSortedResultSet` breaks some workflows. This has been resolved. The old constructor without the `IStatement` argument has been added back.

## Known Issues

An inconsistency between the defaults of `DSIEXT_DATAENGINE_CONTINUE_EXECUTION_ON_ERROR` and `DSI_CONN_STOP_ON_ERROR` causes an assertion failure if the SQLEngine is used in debug mode, and may cause inconsistent behavior in release mode.

The SQLEngine now supports continuing a multi-parameter-set execution after a parameter set prompts an error. This is configured using the `DSIEXT_DATAENGINE_CONTINUE_EXECUTION_ON_ERROR` dataengine property. The default behavior of finishing the execution after an error occurs, and leaving the subsequent parameter sets unused, has been kept.

A new `DSI_CONN_STOP_ON_ERROR` connection property has also been added. Part of the parameter's purpose is to declare whether the DSII will continue on parameter set errors, as described above. The default behavior is to continue after an error occurs. This default retains consistency with the previous behaviour of Java DSII's, where `DSI_CONN_STOP_ON_ERROR` replaced a previous setting, but is inconsistent with the `DSIEXT_DATAENGINE_CONTINUE_EXECUTION_ON_ERROR`'s default.

This inconsistency between the defaults causes an assertion failure if the SQLEngine is used in debug mode, and may also cause unexpected driver behavior if left unchanged. You need to adjust the value of these two properties to match your desired driver behaviour.

The assertion failure may have the following appearance:

```
"simba_abort() called from DSISExtExecutionContext.cpp:43 for
reason:
DSIEXT_DATAENGINE_CONTINUE_EXECUTION_ON_ERROR("N") is
inconsistent with DSI_CONN_STOP_ON_ERROR(0)!"
```

If SDK users see this assertion error or a similar one, the DSIEXT\_DATAENGINE\_CONTINUE\_EXECUTION\_ON\_ERROR and DSI\_CONN\_STOP\_ON\_ERROR are not equivalent and need to be set correctly.

The SDK customer must set the 2 properties equivalent. For example set DSIEXT\_DATAENGINE\_CONTINUE\_EXECUTION\_ON\_ERROR to 'Y' to continue on errors and this will work with the default DSI\_CONN\_STOP\_ON\_ERROR property value.

## Version History

### SimbaEngine 10.1.10

Released 2018-08-31

#### Changes and Resolved Issues

The following is the list of changes and resolved issues in the SimbaEngine X SDK release.

[JDBCClient] In some cases, connections are dropped if assertions are enabled.

### SimbaEngine 10.1.9

Released 2018-08-24

#### Changes and Resolved Issues

The following is the list of changes and resolved issues in the SimbaEngine X SDK release.

[Server] The server returns the following diagnostic under inappropriate circumstances when talking to ODBC clients 10.1.7 or earlier: Execution was deemed to have failed according to value of DSI\_CONN\_ODBC\_PARAMSET\_EXECUTION\_ERROR\_BEHAVIOUR, but client doesn't understand this property.

This has been resolved. The server now suppresses the diagnostic if other warnings or errors have already been posted during execution.

[JDBCClient] Incorrect assertions can be thrown if the client runs with assertions enabled.

[JDBCClient] In some cases, the client fails to connect to servers using 10.1.7 earlier.

[ODBC] A deadlock can occur when the application uses `SQLCancel` on a handle where `async` hasn't been enabled.

[JDBCClient] The client can send multiple `FREE` requests for the same statement. This can lead to the connection state being desynchronized between the client and the server, causing the server to drop the connection.

[JDBCClient] The client fails to close connections properly, causing the connection limit of the server being reached without all connections being utilized. This prevents subsequent connections being made until the unused connections are dropped or time out.

[00132781][ODBC] Driver cannot execute `SQLNumResultCols` on a statement when a previous execution returns `SQL_NO_DATA`.

[ODBC] When using the DataDirect Driver Manager on AIX, drivers search for the shared library under `libodbcinst.so` instead of `odbcinst.so`.

[DSI][Support] The error messages for `InvalidArgument` and `InvalidOperation` exceptions may not be found.

[Support] The `hasMoreRows()` method of `DSIMetadataResultSet` returns true when the cursor is at the last row. This scenario should return false, as there are no more rows are left to fetch.

## SimbaEngine 10.1.8

Released 2018-07-17

### Enhancements and New Features

#### Parser enhancements

[Parser] Support has been added for the following cases:

- UPSERT statements.
- Multi-word datatypes, specifically `DOUBLE PRECISION` for column definition, and `CAST` parameters.
- Top level statements ending with a semicolon.
- Keywords *End*, *Check*, *Percent*, and *Limit* can be used as column names, however they cannot be aliases.

#### Improved fetch performance

[ODBC] Improved fetch performance if application rebinds columns (to the same types) often.

### Updated driver manager detection

[ODBC][00131645] The logic used to detect which driver manager is being used on the system has been updated to support iODBC's library name change (`libiodbcinst.so.2`).

### New parameter set function added

[SQLEngine] The `IExecutionContext::NotifyError()` function allows the DSII to notify the SQLEngine of a parameter set error without posting anything to the warning listener.

### Optimized ASCII data handling

[ODBC] If the DSII returns ASCII data and driver manager encoding is ASCII-compatible, no unnecessary conversion take place.

### Support for specifying whether SQL types signed by default

[Support] The function `SqlTypeMetadataFactory::GetStandardSignednessDefaults()` allows the DSII to control the default sign status for a given SQL type. Under the default implementation, `SQL_TINYINT` is unsigned, and all other types are signed.

### New connection statement limit server property

[Server][ODBCClient][JDBCClient][00131227] The new `CONNSTMTLIMIT` property limits the number of simultaneous statements that the server allows on a given connection.

## Changes and Resolved Issues

The following is the list of changes and resolved issues in the SimbaEngine X SDK release.

[DSI] Added a default implementation for `IStatement/IConnection::GetCustomPropertyType()` that will delegate to `GetCustomProperty()`.

[ODBC][00094579] Changed `SQLConnect` to reject an empty DSN value.

[SQLEngine] Using long data columns in catalog function results can return NULL when the value isn't actually NULL.

[SQLEngine][00131921] `CAST(expr as DECIMAL(x, y))` fails to set the precision to `x`, leaving it at the default.

[JDBCClient][00117250] If 102 or more batches are used in a `PreparedStatement.executeBatch()` call, the call fails and an out-of-bounds array access error occurs.

[JSQLEngine] In some cases, `DSISimpleResultSet.closeCursor()` isn't called as expected.

[CLIDSI][00132160] When using multiple statements on a single connection from multiple threads, a use-after-free error can occur in CLIDSI-based drivers. This can cause the driver to quit unexpectedly, or in possible data corruption.

[DSI] The method `IStatement::ExecuteBatch()` crashes if logging is enabled and set to `DEBUG` or `TRACE` level. This method is invoked (unless it has been overridden by the DSII) when the DSII has been built as a server, and a 10.1.7 or higher JDBCClient uses [https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html#executeBatch\(\)](https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html#executeBatch())

## SimbaEngine 10.1.7

Released 2018-06-22

### Enhancements and New Features

#### Microsoft Visual Studio support

The SDK now supports Microsoft Visual Studio 2017.

#### TimestampAdd and TimestampDiff functions

[00116536][JSQL] The JSQLEngine now supports `TimestampAdd` and `TimestampDiff` scalar functions.

#### Option to disable error checking

[00117019][ODBC] You can now use the DSII to disable checking for correct number of parameters being bound using `DSI_CONN_ODBC_VALIDATE_PARAMETER_COUNT`.

#### New Split method

[Support] Added support for a `Split` method for `simba_strings`.



## Changes and Resolved Issues

The following is the list of changes and resolved issues in the SimbaEngine X SDK release.

[SQLEngine] Driver attempts to compare `TIMESTAMP` to incompatible data types.

This issue has been resolved. Comparisons between SQL `TIMESTAMP` and the following data types are forbidden:

- SQL BIT
- SQL INTEGER TYPES
- SQL DECIMAL
- SQL FLOAT
- SQL REAL
- SQL DECIMAL
- SQL NUMERIC

If a comparison with one of these data types is attempted, an error message will display.

[Server] The server fails to send column information for resultsets unless they are the first result of an execution. Attempting to retrieve a resultset as a subsequent result in an execution fails.

[Server][JDBCClient][JavaDSI] The JDBC client misinterprets parameter set statuses sent back from the server. This issue has been resolved, and logic has been added to the server to compensate for this issue when interacting with older JDBC clients.

[Server] Manually set `LogLevel`, `LogPath`, and `ErrorMessagePath` parameters passed into `SimbaServer_Create` fail to override the parameters in the registry or `.ini` file.

[Support] In some cases, the `SwapFile::FreeSpace` method can cause driver to quit unexpectedly. Note that this method can also be called by the `SwapFile` destructor.

[OLEDB] The OLEDB layer is now lazily-initialized, rather than during static initialization. This change makes initialization behavior consistent with the ODBC layer behavior. The layer is still uninitialized during static destruction.

[SQL] In some cases multi-parameter-set DML executions return `SQL_NO_DATA`, even though one or more parameter sets affected rows.

[SEN-1959] A conversion error can occur during SQL variable-length to SQL fixed-length data types.

[SEN][ODBCClient] The way the client looks for the connection dialog path has changed. The client reads from `ODBCINST.INI` file if there is no entry in the `ODBC.INI` file. The client falls back to looking in Simba Settings if there is no path in either of these locations. The override order, from highest to lowest priority, is now: connection string > DSN > ODBCINST entry > Simba Setting Reader location.

## SimbaEngine 10.1.6

Released 2018-5-25

### Enhancements and New Features

#### **New DSI\_DRIVER\_ALLOW\_CURSOR\_ABANDONMENT property added**

[ODBC] The new `DSI_DRIVER_ALLOW_CURSOR_ABANDONMENT` driver property enables the DSII to optionally allow `SQLExecute`, `SQLExecDirect`, `SQLPrepare`, and any catalog function to be executed during the cursor state without error.

#### **Implemented PIVOT & UNPIVOT**

SQL PIVOT and UNPIVOT relational operators have been added to SQL Engine. SQL Engine parses queries containing these operators and builds the corresponding AE nodes. The DSII layer needs to implement pass-down for these queries.

#### **Support for CREATE TABLE**

[Parser] `PSSql92Generator` now supports CREATE TABLE.

#### **Support for log rotation**

[JavaDSI] `DSILogger` in Java now supports log rotation.

#### **Improved control of connection string data**

[DSI][ODBC] The new method `IConnection::UpdateConnectionSettings` allows the DSII more control over the returned connection strings during connection. This allows the application to check if settings in the connection string are pulled from the registry or read from the user.

#### **Improved handling of metadata**

[DSI] `DSIColumnMetadata` can now be subclassed.

## Improved handling of DML statements

[SQLEngine] Subqueries in DML statements can now contain `ORDER BY` if they also have `TOP` or `LIMIT`. For example, `INSERT INTO T1 SELECT TOP 100 * FROM T2 ORDER BY T2.ID`

## Improved in-memory sorting

[SQLEngine] Drivers can now do an in-memory sort when there is a known limit applied, and that limit is small enough. For example, `SELECT TOP 5 * FROM T1 ORDER BY ID` can be done in-memory, no matter how large `T1` is, as only a maximum of 5 rows need to be kept resident, assuming reasonably-sized columns. This removes all I/O needed by the SQLEngine itself, and improves the efficiency of the sort. This optimization also takes effect if the relation to be sorted is itself known to be limited in size, for example, via `DSIExtResultSet::HasRowCount()/GetRowCount`.

## Improved logging and troubleshooting

[Support] A new version of `simba_abort` has been added that supports printf format strings. This makes it easier to log the cause of a crash, and reduces memory allocation requirements. Wrapper macros `SIMBA_ASSERT` and `SIMBA_RELEASE_ASSERT` have also been added for user convenience.

## New function for SEN-provided logic operators

[00115804][DSI] The function `IConnection::GetDefaultConnStrBehaviour` allows the DSI to control SEN-provided logic operating on the connection string.

## New function for setting signed status of ODBC types

[ODBC] [Support] In previous versions, the function `StatementState::SQLBindParameter` set all SQL types to signed unless the C type is `SQL_C_UBIGINT`, `SQL_C_ULONG`, `SQL_C_USHORT`, and `SQL_C_UTINYINT`. By default, when C type is `SQL_C_UBIGINT`, `SQL_C_ULONG`, `SQL_C_USHORT`, and `SQL_C_UTINYINT`, the application sets the SQL types to unsigned. When C type is `SQL_C_STINYINT`, `SQL_C_SSHORT`, `SQL_C_SLONG`, `SQL_C_SBIGINT`, `SQL_C_TINYINT`, `SQL_C_SHORT`, `SQL_C_LONG`, the application sets the SQL types to signed. All the other types retain their current status.

The new method `SqlCDataTypeUtilities::ChangeSignedness` implements the above default settings. You can override this method to implement your own logic for setting the signed/unsigned status.

### **New parameter for exceptions involving BinaryFile and TextFile constructors**

[Support] A new parameter for `BinaryFile` and `TextFile` constructors controls whether an exception will be thrown if the file cannot be opened.

### **Improved support for multi-parameter-set execution**

The `SQLEngine` now supports continuing a multi-parameter-set execution after a parameter set has an error. This can be configured with the new `DSIEXT_DATAENGINE_CONTINUE_EXECUTION_ON_ERROR` dataengine property. A new `DSI_CONN_STOP_ON_ERROR` connection property has also been added. This parameter declares whether the DSII will continue on parameter set errors, as described above.

### **New IExecutionContext interface**

A new `IExecutionContext` interface has been added, an instance of which is passed to the DSII via `DSIExtSqlDataEngine::OnBeginExecution`. This allows the DSII to be informed of events such as “parameter set started”, ‘parameter set finished’, “parameter set failed”, and “execution finished”. This allows the DSII to delay in checking for errors, even while processing subsequent parameter sets, and so enables batching of updates.

### **New DSI\_CONN\_ODBC\_PARAMSET\_EXECUTION\_ERROR\_BEHAVIOUR connection property**

The connection property `DSI_CONN_ODBC_PARAMSET_EXECUTION_ERROR_BEHAVIOUR` has been added. This allows the DSII to control whether an execution will return `SQL_SUCCESS_WITH_INFO` or `SQL_ERROR` based on which parameter sets succeeded and failed. To support this, there has been a small change to the semantics of `IQueryExecutor::Execute`. The method should only throw an exception if there was some error with the execution as a whole. If there is an error within a parameter set, the DSII will mark that set as failed, and not throw an exception even if *all* parameter sets fail.

### **Updated JDBC escaper**

[JDBCEscaper] The escaper has been extended to also process `LIKE`, `INTERVAL`, and `GUID` escapes.

### **Updated dynamic connection dialog**

[ODBCClient] Users can now skip setting optional fields on the dynamic connection dialog. Also, the connection now fails with an error message when the settings contain an empty but required key name. Previously the Key Name field

on the dialog would be considered a mandatory field, even in cases where it was not required.

### Expanded support for ILogger

[Support] Users can now pass an `ILogger` in when creating a `BinaryFile/TextFile/SwapFile`. The `ILogger` logging macros, such as `INFO_LOG` for example, now optionally accept `ILogger` references as their first parameter. They also now null-check their first parameter, if it is not a reference, and skip logging when it is `NULL`.

### New `IRResults::GetSourceParameterSet` method

[DSI] The method `IRResults::GetSourceParameterSet` has been added. This method is used by the JDBC client to associate execution results with the parameter set which produced them. Unless overridden the JDBC client will assume that results come in the same order as parameter sets, with 1 result per successful parameter set and 0 for failed ones. JNIDSi's implementation assumes that failed parameter sets still produce error results.

### Improved JDBC client/server integration

[SQLEngine] Aggregation of rowcounts/resultsets has been disabled in the context of JDBC (JDBCClient communicating with C++ DSII on server). This has been done because JDBC requires per-parameter-set (batch) results.

### New `DSI_CONN_SUPPORTS_UPDATE_BATCHING` connection property

[JavaDSI][JDBC] The new connection property `DSI_CONN_SUPPORTS_UPDATE_BATCHING` has been added. The previous driver property, `com.simba.dsi.core.utilities.DriverPropertyKey.DSI_SUPPORTS_UPDATE_BATCHING` has been deprecated.

### Support for executing statements in batches

[JDBCClient] The method `Statement.executeBatch` has been added, allowing statements to be run in batches. The call on the client side delegates to the new method `IStatement::ExecuteBatch`. See *Developing Drivers for Data Stores With/Without SQL* for details. This only applies if both server and client are at 10.1.6 or later.

JNIDSi-based drivers attempt to use the existing `IDataEngine.prepareBatch` method. If this fails, the drivers fall back to the `Statement::ExecuteBatch` method.

### Enhancement to `DSISimpleRowCountResult`

[JavaDSI] `DSISimpleRowCountResult` has been extended to allow it to store an unknown "rowcount".

### **New `ErrorException.addSuppressedCustom` and `getSuppressedCustom` methods**

[JavaSupport] The methods `addSuppressedCustom` and `getSuppressedCustom` have been added to `ErrorException`. They are used instead of `Throwable.addSuppressed/getSuppressed`, to support Java 1.6. These methods delegate to the `Throwable` methods in 1.7+ JVMs.

## **Changes and Resolved Issues**

[JDBC] `ResultSetMetaData` for the JDBC `getColumns` API returns the wrong precision and scale for certain columns.

[ODBC] The application fails to detect truncation due to bound `ColumnSize` in some `SQL_C_(W)CHAR` to `SQL_(W)(LONG(VAR))CHAR` conversions when the target encoding is variable-length.

[JDBC] The application fails to restore the original `AutoCommit` state when a transaction that begins with `ITransactionStateListener.notifyBeginTransaction` is rolled back.

[Support] When source values contain leading whitespace in `CHAR` to `TIMESTAMP` conversions, the conversion fails.

[00108858][ODBC] When converting a `NULL SQL_C_(W)CHAR` parameter value which was bound with a very large `ColumnSize`, the driver attempts to allocate a buffer proportional to `ColumnSize`, and fails with an out-of-memory error.

[ODBC] When using `iODBC`, and calling `SQLCopyDesc` with `source == target`, the application clears the source/target.

[JDBCClient][ODBCClient] Boolean values that the JDBC and ODBC clients accept for true are not consistent. This has been resolved. The clients now accept the following values for true: "t", "1", "yes", and "y".

[SQLEngine] When `AEQueryScope` is used as a pointer to out-of-scope objects, the driver may exhibit unexpected behavior.

[00115017][ODBCClient] When the test connection succeeds from the configuration dialog, an incorrect system message icon is displayed.

[SQLEngine] In some cases where queries involved hybrid hash joins on Solaris (SPARC), HP-UX, or AIX platforms, the application would quit unexpectedly.

[Support] Multi-threaded lazy-initialization of a global map can cause the application to quit unexpectedly. The map now initializes during startup.

[00115484][CLIDSJ] In some cases, NULL is incorrectly returned for certain non-NULL values.

[ODBC] When using iODBC the driver allows `SQLGetDiagField` to be called on non-statement handles for the fields `SQL_DIAG_ROW_NUMBER` and `SQL_DIAG_COLUMN_NUMBER`. This behavior violates the ODBC specification. The driver now checks for this situation and returns an error if detected.

[Support] Multiple processes or threads can create swap files with conflicting names. This has been resolved. Swap files created in Windows environments now use a unique GUID for the name of the file.

[Support] Unused swap files are not consistently deleted. This has been resolved. On Windows platforms, the `FILE_FLAG_DELETE_ON_CLOSE` is used to ensure that files are cleaned up if the parent process terminates unexpectedly. On non-Windows platforms, the file is unlinked immediately after being created to ensure that it will not be orphaned and remain on disk.

[Support] `BinaryFile` error checking does not account for partial reads or writes, so `Read/Write` could return incorrect values. This issue has been resolved. `BinaryFile` error checking will now return -1 from `Read/Write` if an error occurs and nothing is read/written.

[Support] `BinaryFile::Close()` closes the `filehandle` even when provided by the `BinaryFile(const simba_wstring&, simba_filehandle)` constructor. This has been resolved. The constructor now more clearly documents that the passed-in handle is `NOT OWN`.

[Support] If the swap file path contains non-Ascii characters, it may not be correctly interpreted when creating a swap file.

[JDBC] In some cases when closing `SStatements` objects, the application incorrectly displays a system message.

[ODBC] The driver does not consistently check for invalid values of `SQL_DESC_PARAMETER_TYPE` in `SQLSetDescField`.

[00115484][CLIDSJ] In some cases `CLIDSJ` provides the DotNet `DSII` with the parameter metadata exposed by the managed `IQueryExecutor` during execution, rather than the metadata bound by the application.

[Support] The behavior of `simba_sprintf` and `simba_vsnprintf` is not consistent across platforms and does not match the expected behavior described in the documentation.

[00115904][ODBCClient] In some cases, when connecting with SSL applications stop responding.

[00115817][ODBCClient] In some cases the ODBC client interprets diagnostics sent from the server as being in the local ANSI encoding rather than UTF-8.

[ODBC] [Support] `ImplParamDescriptorRecord::SetConciseType` skips the call to `SqlTypeMetadataFactory::SetTypeDefaults` if the SQL type has not changed. The ODBC spec says that setting this field should reset associated fields to their defaults.

[Support] In some cases, files opened via `BinaryFile` and subclasses on non-windows platforms are not closed on execution.

[ODBC] Setting a name on a parameter with `SQL_DESC_NAME` fails to clear the `SQL_DESC_UNNAMED` flag on that parameter, preventing the application from binding parameters by name.

[ODBC] In some cases incorrect values are returned from `SQL_ATTR_PARAMS_PROCESSED_PTR` and `SQL_ATTR_PARAM_STATUS_PTR`.

[00116892][DSI] When converting data, and the source encoding is not the target encoding and not the same as `simba_wstring::GetInternalEncoding`, `DSITypeUtilities::OutputConvertedStringData` result includes garbage data following the data to be converted.

[SQLEngine] Using UNION to combine unnamed columns, `SELECT 'a', 'b' UNION SELECT 'c', 'd'` for example, returns empty column names instead of the default `EXPR_XX`.

[SQLEngine] In some cases the TOP operation exposes an incorrect rowcount.

[OLEDB][Server] When run on the server or in the context of OLE DB, `IParameterSetStatusSet::SetStatus` can mask parameter-set errors in some cases.

[00117175][CLIDSI] In some cases when CLIDSI is not using the SQLEngine, memory leaks can occur.

[JDBCClient] In some cases multi-result or batch executions would produce incorrect returns due to the inability of the server to recognize and report back error results during execution.



## SimbaEngine 10.1.5

Released 2018-1-26

### Enhancements and New Features

#### New methods added

- A new `simba_strtok` method has been added. This method is a wrapper for `strtok_r` (POSIX) and `strtok_s` (Windows) functions.
- A new `SQLDataEngine.getSharedSqlConverterGenerator()` method has been added that allows access to the shared `SqlConverterGenerator`.

#### New logging property

[ODBC] The new driver property `DSI_DRIVER_LOG_QUERY_AT_PREPARE` has been added. This allows you to set the log level for logging the query when it is prepared.

#### Error messages

JDBC error messages have been updated with new translations.

#### SSL error messages

Additional error messages regarding SSL have been added to help diagnose connection issues.

#### Customized return values

You can now customize the return value of `DataBaseMetadata.supportsBatchUpdates()` in our JDBC client.

#### Improved column and table logic

The column and table resolution logic has been updated to correctly take `SQL_IDENTIFIER_CASE` and `SQL_QUOTED_IDENTIFIER_CASE` into account.

#### Improved support for Cast function in JSQL

In the JSQLEngine parse tree, `CAST` can now accept Boolean expressions as a Cast operand as defined by SQL 2003 grammar. Also, the cast `TYPE` can now be followed by a parameter list.

## New functions in JSQLEngine

`TIMESTAMPADD` and `TIMESTAMPDIFF` functions are also now supported by the JSQL engine up to the parse tree.

## Log file prefix

Users now have the option to add a prefix to driver log files.

## Support for calculations

SQLEngine now supports `DATE/TIME/TIMESTAMP`, plus/minus `INTERVAL` operations.

## Support for `atod()` method

A new driver property, `DSI_DRIVER_USE_STDLIB_STRTOD`, allows DSII choose whether to use the standard library `strtod()` or the faster but less accurate `atod()` method.

## Setting a default locale

You can now use the `SimbaSettingReader::SetDriverDefaultLocale()` function to set a custom default locale, rather than the default fallback of `en-US`.

## User ID in log files

Log files can now be set to recognize `uid` as sensitive information, and show only as asterisks.

## Support for input passdown on `INSERT` statements

SQLEngine now supports passdown on the input in `INSERT` statements.

## Support for resource optimization

A `Reset()` method has been added to the `Simba::DSI::IQueryExecutor` interface to notify DSII that the cursor is closed. This function will be invoked when `SQLCloseCursor` or `SQLFreeStmt` with the `SQL_CLOSE` option are called.  
[ClientServer]

## Error messaging

Users can now throw a checked exception from the `IReplacer.replace()` method in `JDBCEscaper` using the superinterface for `IReplacer`.

## Changes and Resolved Issues

[Support] `WideStreamConverter` incorrectly converts partial codepoints. This is resolved.

[Support] In some cases drivers can produce incorrect intervals. This is resolved. Checks have been added for invalid interval literals during conversion to interval data types for two missing cases:

- Interval literals that do not contain a value for every field that is implied in the interval qualifier
- Interval literals that do not conform to Gregorian calendar constraints in trailing fields

[Support] Conversions from `C_TIMESTAMP` to `SQL_WCHAR` do not recognize the value set in the `SQL_DESC_PRECISION` field when setting the precision for seconds. This is resolved. If no value is set in the field, the driver records up to 6 digits.

[Support] Errors can sometimes occur when converting `SQL_C_FLOAT` and `SQL_C_DOUBLE` to `SQL_BIGINT`. This issue is resolved.

[ODBC] The `TableName` argument for catalog functions does not recognize escape patterns when checking against the maximum table name length. This issue is resolved.

[ODBC] Error or invalid argument values for `SQLGetStmtAttr`, `SQLSetConnectAttr`, and `SQLBindParameter` are not handled by the program. This issue is resolved.

[ClientServer] Key names are not populated in error messages where parsing of the connection string fails. This issue is resolved.

[Support] During installation, drivers may show a system message regarding creating a temporary swap file name. This issue is resolved.

[ODBC] The `SQL_DESC_DATA_PTR` function is reset after `SQLSetDescField` is called on data type related attributes. This issue is resolved.

[ODBC] When using iODBC as the driver manager, error or invalid values for `SQLSetStmtAttr`, `SQLSetDescField`, `SQLGetFunctions` and `SQLSetConnectAttr` may not be handled correctly. This issue is resolved.

[Support] Conversion from `C_TYPE_TIMESTAMP` to `SQL_CHAR` where precision is not at 9 digits or better produce incorrect results. This issue is resolved.

[Support] Conversion from `SQL_INTERVAL_SECOND` to `C_NUMERIC` that involved fractional seconds produced incorrect results. This issue is resolved.

[Support] The `TypeConverter::toBigInteger` method does not handle `SQL_WCHAR`, `SQL_WVARCHAR`, or `SQL_WLONGVARCHAR` conversions. This issue is resolved.

[Support] The application name output in the logfile is incorrect on MacOS. This issue is resolved.

[SQLEngine] The SQLEngine does not alert users for some invalid SQL queries that select non-aggregated columns that are not referred to in `GROUP BY`. This issue is resolved.

[JSQLEngine] The Identifier class can produce unexpected comparison results due to its reliance on Java's default implementation of `hashCode()` and `equals()`. This is resolved. These two functions are now overridden using strings constructed from the Catalog, Schema and Table names for hashing and comparison.

[ODBC] iODBC does not prevent the application from calling `SQLSetConnectAttr` while there are asynchronously-executing ODBC API functions on child statements of that connection. This is resolved. Logic to check for calls to `SQLSetConnectAttr` while the connection has child statements in the `async` or `NEED DATA` states has been added.

[Support] An undesired null terminator is added to the result of SQL to SQL conversions when converting from float/real/double to char/wchar data types. This issue is resolved.

[Support] An overflow issue sometimes occurs during a C to SQL conversion between interval types, which can cause error checking to be skipped. Errors may also occur when converting fractional seconds in various interval to interval conversions. This issue is resolved.

[JSQLEngine] Issues with `DSIExtJResultSet`s returned from `SqlDataEngine.openTable()` are not being communicated to users. This issue is resolved.

[ODBCClient] System alerts during conversion of retrieved results are ignored; this can cause the application to inadvertently read invalid values without alerting the user. This issue is resolved.

[ODBC] iODBC allows users to set `SQL_ATTR_ENABLE_AUTO_IPD` as a connection attribute in ODBC 3.X mode. This issue is resolved.

[JDBC Client] Service entries of `java.sql.Driver` type are missing from client jars. This prevents the Auto-Loading feature for JDBC client from functioning. This issue is resolved.

[Support] The function `EncodingInfo::GetMaxCodeUnitsInCodePoint()` may not return the correct value for encoding type `ENC_EUC_JP` and `ENC_KANJI_EUC_OU_TD`. This can lead to miscalculated column length and unexpected truncation errors. This issue is resolved.

[SQLEngine] Values lists that include NaNs may not sort correctly. This is resolved. NaN is now always treated as the largest value in the list.

[Simba.Net] Users may not be able to properly access the driver properties `DSI_DRIVER_LOG_QUERY_AT_PREPARE` and `DSI_DRIVER_USE_STDLIB_STRTOD` from `CLIDSI`. This issue is resolved.

[Support] Under some circumstances target buffers cannot hold the source data of `CInterval` data type, which results in a memory corruption issue. This issue is resolved.

[ODBC] Calling `SQLDescField SQL_DESC_BIND_TYPE` and a `NULL ValuePtr` can cause the application to quit unexpectedly. This issue is resolved.

[SEN] Under some circumstances the `CToSql` converter may be created before metadata is properly set. This issue is resolved.

[ODBC] When `SQLGetData()` is called with an insufficient buffer it returns truncated data as expected, but also consume the rest of the data so that following calls to `SQLGetData()` return no data. This issue is resolved.

[API] The description for `GenerateCatScalarFn` incorrectly stated that the parameter list requires exactly 2 children. This is resolved. The description now reads that it requires at least 2 children.

[SQL Engine] Issues could result when clients did not implement `Execute` and `RetrieveData` of `DSIExtScalarFunction.h`. This is resolved. Default implementations of `Execute` and `RetrieveData` of `DSIExtScalarFunction.h` now alert the user if the client doesn't implement them.

[ODBC] Users are not alerted when the result for `SQLRowCount` is truncated. This issue is resolved.

[JavaSupport] Commas in the argument list of the function `JDBCESaper` may not be correctly parsed. This issue is resolved.

[ODBC] On Solaris Sparc, `SQLGetRowCount()` may return incorrect results when `ROW_COUNT_UNKNOWN` is expected. This issue is resolved.

[ODBC] The error message `StrRightTruncErr` cannot be retrieved under certain conditions. This issue is resolved.