

# Matisse<sup>®</sup> 8.3.2

## Release Notes

December 2009



Matisse 8.3.2 Release Notes

Copyright ©1992–2009 Matisse Software Inc. All Rights Reserved.

Matisse Software Inc.  
930 San Marcos Circle  
Mountain View, CA 94043  
USA

Printed in USA.

This manual is copyrighted. Under the copyright laws, this manual may not be copied, in whole or in part, without prior written consent of Matisse Software Inc. This manual is provided under the terms of a license between Matisse Software Inc. and the recipient, and its use is subject to the terms of that license.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. and international patents.

TRADEMARKS: Matisse and the Matisse logo are registered trademarks of Matisse Software Inc. All other trademarks belong to their respective owners.

PDF generated 6 January 2010

# Contents

<b>1</b>	<b>New Features in Matisse 8.3</b>	<b>5</b>
1.1	Overview	5
1.2	Enterprise Manager Tool	5
	Schema Viewer	5
	SQL Analyzer	5
1.3	Matisse Data Transformation Services	6
	NULL export format	6
	Bytes Qualifier	6
	Media Data	6
1.4	Database Utility Commands	7
	mt_server monitor	7
	mt_connection count	7
1.5	Matisse SQL	7
	ORDER BY with Navigation	7
	GROUP BY with Navigation	7
	DISTINCT with Navigation	8
	SELECT FILTERED	8
	SELECT UNFILTERED	9
	SELECT PARALLEL	9
	SQL Methods in Projection	10
	String Built-in Functions	10
	Numeric Built-in Functions	11
	Bitwise Operators	12
	Reports demo	12
1.6	.NET Binding	12
	.NET 3.5 Support	12
	Generated Code	12
	Generics	13
	LINQ	13
	Examples	13
1.7	Java Binding	13
	Generics	13
	Generated Code	13
	Examples	13
1.8	Database Configuration Parameters	14
	MAXSQLDOP	14
	MAXSQLDOP	14
1.9	License Key Format	14
<b>2</b>	<b>Compatibility with Previous Releases</b>	<b>15</b>
2.1	Matisse 8.3 Data Migration	15

Step 1 .....	15
Step 2 .....	15
2.2 Client Connections .....	15
2.3 License Key Format .....	15
Release 8.3 .....	15
<b>3 Platform-Specific Topics .....</b>	<b>16</b>
3.1 Linux .....	16
3.2 MacOS .....	16
3.3 Solaris .....	16
3.4 Windows .....	16
<b>4 Update History .....</b>	<b>17</b>
Resolved in Matisse 8.3.2 .....	17
Resolved in Matisse 8.3.1 .....	17
Resolved in Matisse 8.3.0 .....	18
<b>5 Documentation .....</b>	<b>20</b>
Matisse documents available on the Web .....	20
Documents included with Matisse standard installation .....	20
Open source bindings .....	20

# 1 New Features in Matisse 8.3

## 1.1 Overview

The Matisse 8.3 release introduces new features and major enhancements in the Matisse product line:

- The *Matisse Enterprise Manager* has been enhanced to improve the developers experience. The Schema Viewer has been extended to ease access to detailed representation of Matisse schema objects. The SQL analyzer editor has been extended to improve the readability of SQL statements and methods.
- *Matisse Data Transformation Services* provide new features that improve the migration of data coming from legacy relational databases.
- Some of the Matisse command line utilities have received new options to ease database administration.
- The new SQL features greatly enhance ad-hoc reporting capabilities while improving the overall performance of `SELECT` statements. This release also introduces support for parallel processing of queries.
- Matisse .NET binding has been upgraded to support Microsoft's .NET Framework 3.5 with the addition of a LINQ provider.
- Matisse Java binding has been upgraded to support Java Generics.

## 1.2 Enterprise Manager Tool

The *Matisse Enterprise Manager* has been enhanced to improve the developers experience.

**Schema Viewer** The Schema Object Viewer displays the Matisse Meta-schema in the tree hierarchy similar to the one used to display the application schema. In addition, the `View Details` popup menu has been added to the summary tables for Classes, Methods, Indexes and Entry Point Dictionaries to directly access the schema object description from the summary table.

**SQL Analyzer** The SQL analyzer editor provides an option to show or hide the source code line number combined with a faster SQL syntax highlighting to improve the readability of SQL statements and methods.

## 1.3 Matisse Data Transformation Services

*Matisse Data Transformation Services* provide new features that improve the migration of data coming from legacy relational databases.

### NULL export format

The new `NullSymbol` option tag allows you to export `NULL` values with the keyword of your choice or with an empty field. The `NullSymbol` option has been added to the XML Export options descriptor. The default value for `NullSymbol` option is `NULL`. To export `NULL` values as empty, define the tag as follows:

```
<?xml version="1.0"?>
<DTOptions>
  <DTSexport>
    <nullSymbol></nullSymbol>
  </DTSexport>
</DTOptions>
```

### Bytes Qualifier

The new `BytesQualifier` option tag allows you to qualify a media type represented in hexadecimal coded-ascii characters. The `BytesQualifier` option has been added to the XML Import and XML Export options descriptors. The default value for `BytesQualifier` option is `0x`. To export media data values with no qualifier, define the tag as follows:

```
<?xml version="1.0"?>
<DTOptions>
  <DTSexport>
    <bytesQualifier></bytesQualifier>
  </DTSexport>
</DTOptions>
```

### Media Data

The new `MediaData` option tag allows you to import/export large binary data (`MT_BYTES`, `MT_IMAGE`, `MT_AUDIO`, `MT_VIDEO`) and large text data (`MT_TEXT`) embedded as field values in the CSV file or externalized in files with a filename referred in the field. The `MediaData` option has been added to the XML Import and XML Export options descriptors. The `MediaData` option has two possible values `File` or `Column`. The default value is `File`. To import media data values embedded in the CSV file, define the tag as follows:

```
<?xml version="1.0"?>
<DTOptions>
  <DTSexport>
    <mediaData>Column</mediaData>
  </DTSexport>
</DTOptions>
```

For convenience `-media File|Column` has also been added to the `mt_dts` command line options.

## 1.4 Database Utility Commands

Some of the Matisse command line utilities have received new options to ease database administration.

### mt\_server monitor

The `mt_server` command with the `monitor` option allows you to monitor the server activity in a command line window. This command provides information about the database state, database objects count, memory usage and disk I/O activities.

```
mt_server [OPTIONS] monitor [-riwh]
-r, --request Number of requests before exiting (0 unlimited,
default 4)
-i, --interval Refresh interval in seconds (default 3)
-w, --wide Display for wide screen (120 columns)
-h, --help Display this help and exit
```

### mt\_connection count

The `mt_connection` command with the `count` option returns the number of active connections. The count includes the connection for the `mt_connection` command itself, so the minimum value returned by this command is always 1.

## 1.5 Matisse SQL

The new SQL features greatly enhance ad-hoc reporting capabilities while improving the overall performance of SELECT statements.

### ORDER BY with Navigation

The `ORDER BY` clause can now also sort objects accessed via a relationship according to the values of some of their attributes. To sort objects in navigational queries, you need to specify the navigation path in the `ORDER BY` clause as follows:

```
SELECT m.title, m.starring.lastName AS Starring
FROM movie m
ORDER BY m.title, m.starring.lastName DESC;
```

### GROUP BY with Navigation

You can now group by objects in navigational queries by specifying the navigation path in the `GROUP BY` clause. For example, the following statement lists the number of contracts per contract type, for each employee position in each department:

```
SELECT
    d.DepartmentName,
    d.employees.Class_Name AS Position,
    d.employees.Contract,
    count(d.employees.*)
FROM
    Department d
```

```
GROUP BY
    d.DepartmentName,
    d.employees.Class_Name,
    d.employees.Contract
ORDER BY
    d.DepartmentName,
    d.employees.Class_Name DESC,
    d.employees.Contract ASC;
```

## DISTINCT with Navigation

`SELECT DISTINCT` has been extended to support relationship navigation. To retrieve distinct values in navigational queries, you need to specify the navigation path for each `DISTINCT` property in the `ORDER BY` clause as follows:

```
SELECT DISTINCT
    d.DepartmentName,
    CONCAT(' ', CONCAT(d.employees.Contract, ' ')) AS
    Contract,
    CAST((d.employees.Salary / 12) AS INT) AS Monthly
FROM
    Department d
ORDER BY
    d.DepartmentName,
    d.employees.Contract,
    d.employees.Salary DESC;
```

## SELECT FILTERED

The `SELECT FILTERED` statement applies the relationship navigation filters in the `WHERE` clause to the relationship navigation in the `Select-list`. The SQL projection produced is equivalent to the SQL projection of a SQL relational equi-join.

The following statement returns only the rows matching all the conditions expressed in the `WHERE` clause that is only 2 rows:

```
SELECT FILTERED
    d.DepartmentName,
    d.employees.EmpId,
    d.employees.Salary
FROM
    Department d
WHERE
    d.employees.Salary = 180000;
```

DepartmentName	EmpId	Salary
Finance	3	180000
Finance	19	180000

The result of the `SELECT FILTERED` statement is equivalent to the result of a `SELECT` statement with a `HAVING` clause that includes the navigation predicates defined in the `WHERE` clause of the `SELECT FILTERED`.

```

SELECT
    d.DepartmentName,
    d.employees.EmpId,
    d.employees.Salary
FROM
    Department d
HAVING
    Salary = 180000;

```

DepartmentName	EmpId	Salary
-----	-----	-----
Finance	3	180000
Finance	19	180000

## SELECT UNFILTERED

The `UNFILTERED` keyword forces a direct `SELECT` statement to build the full SQL projection on the server-side the same way a block statement or a SQL Method would do it, thus eliminating the need for returning objects to the client workspace.

```

SELECT UNFILTERED
    d.DepartmentName,
    d.employees.EmpId,
    d.employees.Salary
FROM
    Department d
ORDER BY
    d.DepartmentName;

```

The SQL statement above is equivalent to the same `SELECT` statement executed in a block statement:

```

BEGIN
    SELECT
        d.DepartmentName,
        d.employees.EmpId,
        d.employees.Salary
    FROM
        Department d
    ORDER BY
        d.DepartmentName;
END;

```

## SELECT PARALLEL

The `PARALLEL` query hint specifies the degree of parallelism requested for the execution of a SQL statement. The actual number of threads used by a parallel query is determined at query plan execution initialization and is determined by the degree of parallelism and number of threads available in the SQL parallel processing pool of threads. The maximum degree of parallelism is set at the server level and defines the upper value

which determines the maximum number of threads that are being used. You need to set the appropriate database configuration parameters to enable and to control the resources dedicated to parallel processing of queries.

```
SELECT PARALLEL (4)
    d.DepartmentName,
    d.employees.EmpId,
    d.employees.LastName,
    d.employees.GetAccrualsName(Bank::ListBankName(FALSE)) AS "Bank
Name",
    d.employees.GetAccrualsQuantity(Bank::ListBankName(FALSE)) AS
Total
FROM
    Department d
ORDER BY
    d.DepartmentName,
    d.employees.LastName;
```

## SQL Methods in Projection

The Select-list can now include SQL method calls. The SQL statement below uses both the instance methods `GetAccrualsName()` and `GetAccrualsQuantity()` defined on the `Employee` class and the static method `ListBankName()` defined on the `Bank` class to list the negative accruals for the Legal department employees:

```
SELECT
    d.DepartmentName,
    d.employees.EmpId,
    d.employees.LastName,
    d.employees.GetAccrualsName(Bank::ListBankName(FALSE)) AS "Bank
Name",
    d.employees.GetAccrualsQuantity(Bank::ListBankName(FALSE)) AS
Total
FROM
    Department d
WHERE
    d.DepartmentName = 'Legal'
HAVING
    Total < 0
ORDER BY
    d.DepartmentName,
    d.employees.LastName;
```

## String Built-in Functions

The new built-in functions to manipulate character string values provide a variety of solutions to format data in ad-hoc reports.

Function	Description
LEFT	Returns the leftmost N characters from a string
LOCATE	Returns the position of the first occurrence of a substring in a string
LPAD	Returns a string left-padded with padstring
REPLACE	Returns a string with all occurrences of the fromstring replaced by tostring
REPLICATE	Returns a string consisting of the string repeated N times.
REVERSE	Returns a string with the order of the characters reversed.
RIGHT	Returns the rightmost N characters from a string
RPAD	Returns a string right-padded with padstring
TRIM	Returns a string with leading and trailing characters that appear in trimstring removed.

## Numeric Built-in Functions

New built-in functions to manipulate numeric values have been added to further extend SQL programming capabilities for applications that require in depth analysis of complex data.

Function	Description
BIT_COUNT	Return the number of bits that are set
ABS	Returns the absolute value
ACOS	Returns the arc cosine
ASIN	Returns the arc sine
ATAN	Returns the arc tangent
ATAN2	Returns the arc tangent of the two variables
CEILING	Returns the smallest integer value not less than the number
COS	Returns the cosine
COT	Returns the cotangent
DEGREES	Returns the number converted from radians to degrees
EXP	Returns the value of e (the base of natural logarithms) raised to the power of the number
FLOOR	Returns the largest integer value not greater than the number
LN	Returns the natural logarithm
LOG10	Returns the base-10 logarithm

Function	Description
LOG2	Returns the base-2 logarithm
LOG	Returns the logarithm of a number to the base B
PI	Returns the value of pi
POWER	Returns the value of X raised to the power of Y
RADIANS	Returns the number converted from degrees to radians
ROUND	Rounds the number to D decimal places.
SIGN	Returns the sign of the number
SIN	Returns the sine
SQRT	Returns the square root of a non negative number
TAN	Returns the tangent
TRUNCATE	Returns the number truncated to D decimal places

## Bitwise Operators

The new bitwise operators can be used in both the `SELECT` list and the `WHERE` clause as well as in block statement and SQL methods.

Operator	Meaning
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
~	Invert bits
<<	Left shift
>>	Right shift

## Reports demo

Matisse installations now include a new demo (Reports) which demonstrates Matisse SQL-based reporting capabilities. This demo presents ad-hoc SQL queries taking full advantage of Matisse relationship capabilities and Object-oriented SQL programming.

## 1.6 .NET Binding

### .NET 3.5 Support

Matisse .NET binding has been upgraded to support Microsoft's .NET Framework 3.5.

### Generated Code

The class definition for the persistent classes uses the **partial** keyword modifier so developers can more easily extend the generated source code. The generated persistent classes also includes additional methods.

**Generics** Matisse .NET binding has been upgraded to support Generics.

**LINQ** Matisse .NET binding now includes a LINQ provider. The Matisse Linq provider is included in an additional library named MatisseNet.Linq. The example below illustrates how to retrieve views from objects using a LINQ query with navigation.

```
// Get the database Entity Provider
LinqExample mdb = new TestLinq.Schema.LinqExample(conn);

// Query Expression
var projview2 = from p in mdb.Projects where p.Budget < 250
                select new { Project = p.ProjectID,
                             Manager = p.ManagedBy.EmpId,
                             City = p.ManagedBy.Address.City };
```

**Examples** The .NET binding examples have been updated to further reduce the programming learning curve and to enhance the developers experience. The binding examples include new projects which demonstrates Connection Pooling, Class Reflection and LINQ query expression with Matisse.

## 1.7 Java Binding

**Generics** Matisse Java binding has been upgraded to support Java Generics.

**Generated Code** The Java source code generated for persistent classes includes additional methods. The class source code includes two methods to enumerate the class own instances (excluding subclasses) and a method returning the count of own instances (excluding subclasses).

```
public static <E extends com.matisse.reflect.MtObject>
    com.matisse.MtObjectIterator<E>
        ownInstanceIterator(com.matisse.MtDatabase db)
public static <E extends com.matisse.reflect.MtObject>
    com.matisse.MtObjectIterator<E>
        ownInstanceIterator(com.matisse.MtDatabase db, int
            numObjPerBuffer)
public static long getOwnInstanceNumber(com.matisse.MtDatabase db)
```

The generated code also includes new checking methods for each attribute:  
 MyClass.is<MyAttribute>Null() (only for nullable attribute)  
 MyClass.is<MyAttribute>DefaultValue()

**Examples** Matisse Java binding examples have been redesigned to further reduce the programming learning curve and to enhance the developers experience. The Java binding examples include new projects which demonstrates Connection Pooling, Class Reflection and SQL Methods in Java.

## 1.8 Database Configuration Parameters

New optional database parameters have been added to support parallel processing of queries.

**MAXSQLDOP** This parameter defines the maximum degree of parallelism which determines the maximum number of threads that are being used. Specifying a value of 0 disables parallel processing. The maximum value of `MAXSQLDOP` is the number of logical CPUs on the server. The default value of the `MAXSQLDOP` parameter is 0.

**MAXSQLTHRDPOOL** This parameter defines the maximum number of threads in the pool of threads dedicated to parallel processing of SQL queries. Specifying a value of 0 disables parallel processing. The maximum value of `MAXSQLTHRDPOOL` is twice the number of logical CPUs on the server. The default value of the `MAXSQLTHRDPOOL` parameter is 0.

## 1.9 License Key Format

The customer license key format has changed in release 8.3. The key can now include an OEM product identifier to help OEM customers manage multiple product lines. Matisse 8.3 does not recognize license keys issued for prior releases.

## 2 Compatibility with Previous Releases

### 2.1 Matisse 8.3 Data Migration

Matisse Server 8.3 comes with several changes in the data format. You must use the `mt_xml` tool for converting an existing database (8.2.x or prior) into the 8.3 format.

#### Step 1

Before installing 8.3.x, save your schema in ODL and your data in XML format:

```
mt_sdl -d <dbname> export -odl schema.odl
mt_xml -d <dbname> export -f data.xml -full
```

Prior to 8.0.4, use the command below:

```
mt_xml -d <dbname> -xml data.xml -full
```

You may check the [Matisse XML Programming Guide](#) for more options with exporting in XML format.

#### Step 2

You may now install Matisse 8.3.x on your machine and then restore the schema and the data as follows:

```
mt_sdl -d <dbname> import -odl schema.odl
mt_xml -d <dbname> import -f data.xml
```

### 2.2 Client Connections

Only 8.3.x clients may be used with 8.3.x servers.

The clients for earlier releases of Matisse are incompatible with the 8.3.x server. Consequently, you must upgrade any older clients to 8.3.x before attempting to access an 8.3.x server.

### 2.3 License Key Format

#### Release 8.3

The customer license key format has changed in release 8.3. Matisse 8.3 does not recognize license keys issued for prior releases. Upon installation of Matisse 8.3, a license key with limited features is automatically issued.

## 3 Platform-Specific Topics

### 3.1 Linux

The following Linux distributions on x86 (32-bit) and x86\_64 (64-bit) chips families are supported:

- Red Hat Enterprise Linux 4.x / 5.x
- Fedora Core 3 up to 9
- SUSE Linux Enterprise Server 9 / 10
- SUSE 9.x / 10.x
- CentOS 4.x / 5.x

Any other Linux distributions, where Matisse 8 has not been tested, require Linux kernel 2.6.9 on systems based on x86 (32-bit) or x86\_64 (64-bit) chips families.

### 3.2 MacOS

The MacOS X version for Intel of Matisse DBMS is available upon request.

### 3.3 Solaris

Support for Solaris 10 on SPARC with 32-bit kernel and 64-bit kernel.

Support for Solaris 10 on x86 (32-bit) and AMD64 (64-bit) chips families.

### 3.4 Windows

Support for Windows (2000/XP/2003/2008/Vista/7) on systems based on x86 (32-bit) and x86\_64 (64-bit) chips families.

## 4 Update History

This section contains the list of bug fixes and minor feature changes between releases. You may refer to it before upgrading to see if the new release resolves a known problem or adds a needed feature.

### Resolved in Matisse 8.3.2

- On Linux 32-bit, the `mt_xml` utility may fail to import XML files larger than 2Giga-bytes.
- In the Java binding, the JDBC driver may fail to execute navigational select statements with the error `MATISSE-E-MULTIPLYDEFINEDPROP`, Multiple definitions exist for property "myrel".
- In SQL, `SELECT` statements support parallel processing. A `SELECT PARALLEL(n)` query hint specifies the degree of parallelism requested for the execution of a SQL statement.

### Resolved in Matisse 8.3.1

- Matisse Enterprise Manager Lite, the Enterprise Manager for Matisse embedded databases, is now included in the Matisse Lite package.
- In the Java binding, new methods have been added to `MtStatement` and `MtResultSet` classes which improve the control of object fetching from the server when enumerating objects from a query result set.

```
MtResultSet.getFetchObjectNumber()
MtResultSet.setFetchObjectNumber(int fetchObjNum)
MtStatement.getFetchObjectNumber()
MtStatement.setFetchObjectNumber(int fetchObjNum)
```

- In the .NET binding, the C# Reflection example has been extended to demonstrate adding and removing attributes, relationships and indexes.
- The .NET binding examples include a new project which illustrates executing queries using Matisse LINQ provider.
- In the .NET binding, new accessors have been added to `MtCommand` and `MtDataReader` classes which improve the control of object fetching from the server when enumerating objects from a query result set.

```
MtDataReader.FetchObjectNumber
MtCommand.FetchObjectNumber
```

- In some cases, the XML import procedure may fail to report the exact number of objects loaded into the database.
- The `mt_xml` utility returns a misleading error message when loading a XML document referring to an external media file that does not exist.  
`MTXML-E-MEDIAFILEERROR`, Unable to export file image\_123.blb: No such file or directory
- In Matisse Enterprise manager SQL Analyzer the execution of a `'SELECT UNFILTERED/FILTERED * FROM ...'` statement may fail with the `MATISSE_OBJECTNOTFOUND` error.

- In SQL PSM, the new `REMOVE_DUPLICATES()` method has been added to the `SELECTION` object to remove the duplicate objects contained in a selection.
- In SQL, a select statement with a select list containing only constants returns one row with the constants even when the `WHERE` clause is false. (i.e. `SELECT 1, true FROM MtObject where 1 = 2`).
- In SQL, in some cases a select statement with a `WHERE` clause using multiple SQL Methods and a `OR` condition to filter objects may return a selection with duplicate objects.
- In SQL PSM, in some cases accessing a relationship in a `FOR` loop may return `NULL` while the relationship is not empty.
- In SQL PSM, a `MATISSE_SQLRUNTIMEERR` error may be returned when accessing a relationship with a `NULL` object.
- In SQL, a statement including an unsupported built-in function may report an unspecific compilation error.
- SQL does not support automatic cast of Timestamp into Date when comparing a property of type Date with a constant of type Timestamp.
- In SQL, a `SELECT DISTINCT` statement with the `OID` column fails to compile with the `MATISSE_OBJECTNOTFOUND` error.
- In SQL, in some cases the `ORDER BY OID ASC` clause does not return a correctly sorted result.
- In SQL, in some cases the enumeration of objects from a query result set does not fully optimized the object fetching from the server.
- The `mt_odl` utility may fail to delete multiple relationships in multiple classes with inheritance from the database schema when loading an ODL file where these relationships are missing.
- The .NET Object Manager utility `mt_dnom` may fail to generate the C# source code for SQL methods returning a `TABLE` with columns of type object.
- The utility `mt_sdl` may fail to generate the Java source code for SQL methods returning a `TABLE` with columns of type object.

### Resolved in Matisse 8.3.0

- Matisse Enterprise Manager now displays the Matisse Meta-schema in the tree hierarchy similar to the one used to display the application schema.
- In Matisse Enterprise Manager Schema, the `View Details` popup menu has been added to the summary tables for Classes, Methods, Indexes and Entry Point Dictionaries to directly access the schema object description from the summary tables.
- Matisse Data Transformation Services (`mt_dts`) has now a new option tag to import/export large binary data (`MT_BYTES`, `MT_IMAGE`, etc.) and large text data (`MT_TEXT`) as field values. The `MediaData` option has been added to the XML Import and XML Export options descriptor. The

MediaData option has two possible values File or Column. The default value is File:

```
<mediaData>File</mediaData>
```

For convenience `-media File|Column` has also been added to `mt_dts` command line options.

- Matisse Data Transformation Services (`mt_dts`) has now a new option tag to qualify a media type represented in hexadecimal coded-ascii characters. The `BytesQualifier` option has been added to the XML Import and XML Export options descriptor. The default value for `BytesQualifier` option is `0x`.

```
<bytesQualifier>0x</bytesQualifier>
```

- Matisse Data Transformation Services (`mt_dts`) has now a new options tag to export `NULL` values with the keyword of your choice or to an empty field. The `NullSymbol` option has been added to the XML Export options descriptor. The default value for `NullSymbol` option is `NULL`:

```
<nullSymbol>NULL</nullSymbol>
```

- In SQL, new built-in functions to manipulate character string values have been added including `REPLACE()`, `REVERSE()`, `REPLICATE()` and many more.
- In SQL, new built-in functions to manipulate numeric values have been added including `SIN()`, `SQRT()`, `TAN()`, `TRUNCATE()`, `ROUND()` and many more.
- In SQL, new operators (`&` | `^` | `~` | `%` | `<<` | `>>`) to manipulate bits have been added.
- In SQL, a statement including the number `-9223372036854775808` fails to compile with the `'MATISSE SQL constant too long'` error.
- In SQL, in some cases the comparison with numeric values near `INT64 MIN` (`-9223372036854775808`) or `INT64 MAX` (`9223372036854775807`) may fail to return the correct result.
- In SQL, the `LIKE` predicate in a `WHERE` clause does not support an expression, but only a text constant or a variable when used in a block statement.
- In SQL, `NTEXT` and `NCHAR(n)` are not supported as synonyms to respectively `TEXT CHARACTER SET UTF16` and `NVARCHAR(n)` to describe unicode character string types.
- Exporting into XML files a database with very large relationships may cause the `mt_xml` process to grow abnormally in size.
- In the Java binding the `skip(int num)` method defined on the `MtObjectIterator` class called with `num` equal to `0` behaves as if `num` was set to `1`.

## 5 Documentation

### Matisse documents available on the Web

The following documents are available at <http://www.matisse.com/developers/documentation>:

- Installation guides for Linux, MS Windows, and Solaris
- *Getting Started with Matisse*
- *Matisse SQL Programmer's Guide* (includes user's guide for `mt_sql`)
- *Matisse .NET Programmer's Guide* (and example applications)
- *Matisse Java Programmer's Guide* (and example applications)
- *Matisse C++ Programmer's Guide* (and example applications)
- *Matisse C API Reference*
- *Matisse ODL Programmer's Guide*
- *Matisse Rose Link User's Guide*
- *Matisse Server Administration Guide*
- *Matisse XML Programming Guide* (includes user's guide for `mt_xml`)
- *Matisse Data Transformation Services Guide* (includes user's guide for `mt_dts`)
- *Matisse Editor User Guide for MS Windows* (user's guide for `mt_editor`)
- *Matisse Editor User Guide for X/Motif* (user's guide for `mt_editor`)

### Documents included with Matisse standard installation

- Guide to Matisse documentation and other resources: `readme.html`
- *Matisse .NET Binding API Reference*: `docs/NET/Solution_MatisseNet.htm`
- *Matisse Java Binding API Reference*: `docs/java/api/index.html`
- *Matisse C++ Binding API Reference*: `docs/cxx/api/index.html`

### Open source bindings

- *Matisse Eiffel Programmer's Guide*
- *Matisse Perl Programmer's Guide*
- *Matisse PHP Extension Reference*
- *Matisse Python Interface Reference*

The Matisse Smalltalk documentation is included in the Matisse Smalltalk binding package.