

Matisse[®] 7.0.9

Release Notes

1st Edition

October 2005



Matisse 7.0.9 Release Notes

Copyright ©1992–2005 Matisse Software Inc. All Rights Reserved.

Matisse Software Inc.
433 Airport Blvd
Burlingame, CA 94010
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 25 September 2001

Contents

1	New Features in Matisse 7.0.6	4
1.1	Overview	4
1.2	New Database Tools	4
	Matisse Enterprise Manager	4
	mt_sdl	4
	mt_connection	4
	mt_transaction	4
1.3	Schema Changes	5
	Mono-Directional Relationships	5
1.4	Matisse SQL	5
	LIMIT, OFFSET for SELECT statement	5
	Get a Successor at a Position	5
	UPDATE SELF in SQL Method	6
	Generalized Method Invocation	6
	Enhancements	6
	New C-API MtSQLParse	6
	MOD built-in function	6
1.5	.NET Binding	7
1.6	Java Binding	7
2	Compatibility with Previous Releases	8
2.1	Data Migration	8
	Step 1	8
	Step 2	8
2.2	Client Connections	8
2.3	Java Binding	8
3	Platform-Specific Topics	9
3.1	Solaris	9
	Solaris 64-bit Kernel and Solaris on Intel	9
3.2	FreeBSD	9
3.3	Linux	9
4	Update History	10
5	Documentation	13

1 New Features in Matisse 7.0

1.1 Overview

The Matisse 7.0 release introduces two major features in the Matisse product line:

- The *Matisse Enterprise Manager* is now the primary user interface to Matisse.
- The *SQL Stored Methods* (SQL PSM) capability is fully completed and allows you to build reusable SQL components within the database server.

The Matisse 7.0 release also includes many improvements in SQL, indexing, language bindings, and the Matisse client library.

1.2 New Database Tools

Matisse Enterprise Manager

The *Matisse Enterprise Manager* regroups in a single tool: the management of database servers, the management of database schemas, import and export of data from relational databases and XML documents, as well as various security and administration functions. It also includes a SQL analyzer tool to help optimize complex queries and produce data result-sets in a table format.

Matisse Enterprise Manager replaces the former DBA tool which has been discontinued.

The functionality provided by the following commands is also available from the Enterprise Manager. These commands are provided so that Matisse can be administered without a graphical environment, and also for writing administration scripts.

The Matisse 7.0.6 release has enabled the backup feature to the Enterprise Manager.

mt_sdl

The `mt_sdl` utility replaces the former `mt_odl` utility. With `mt_sdl` you can import and export schemas into or from your database. You can also generate stub classes for building Java, C++ or Eiffel applications.

mt_connection

The `mt_connection` utility allows you to view connections and kill active connections.

mt_transaction

With the `mt_transactionl` utility, you can view transactions, enable or disable transaction processing, and abort pending transactions.

1.3 Schema Changes

Mono-Directional Relationships

You can now define relationships in your schema without an inverse relationship. This simplifies schema design when an inverse relationship is not required. For instance in the following SQL DDL statement, the relationship `Team` is created with an inverse `ReportsTo`.

```
CREATE CLASS Manager UNDER Employee (
    Team REFERENCES SET (Employee)
    INVERSE Employee.ReportsTo
);

ALTER CLASS Employee ADD RELATIONSHIP
    ReportsTo REFERENCES (Manager)
    INVERSE Manager.Team;
```

If you do not need the inverse `ReportsTo`, you can do more simply:

```
CREATE CLASS Manager UNDER Employee (
    Team REFERENCES SET (Employee)
);
```

Mono-directional relationships are also available with ODL and UML (Matisse Rose Link).

The same schema can be written in ODL:

```
interface Manager : Employee : persistent {
    relationship Set<Employee> Team;
};
```

1.4 Matisse SQL

LIMIT, OFFSET for SELECT statement

A new syntax has been added to specify the maximum number of objects to return, and to specify the number of objects to skip before starting to return objects:

```
SELECT p.Name FROM Project p ORDER BY p.Budget LIMIT 10 OFFSET 5;
```

Note that the `SET MAXOBJECTS` statement has been dropped.

Get a Successor at a Position

A new syntax has been added to get a successor object at a specified position in a relationship. When a relationship name is followed by a pair of parentheses with an argument, it returns an object at the specified position the relationship. For example, the following example returns the first member's name in each `Project` object which has a relationship `Members`:

```
SELECT p.Members (1).Name FROM Project p;
```

UPDATE SELF in SQL Method

In SQL instance methods, you can use UPDATE SELF statement to update the current instance. For instance, the following SQL method updates a customer's address:

```
CREATE METHOD ChangeAddress(newAddr VARCHAR, newCity VARCHAR, ...)
RETURNS NULL
FOR Customer
BEGIN
    UPDATE SELF SET Address = newAddr,
                  City = newCity;
    ...
END;
```

Generalized Method Invocation

You can call methods defined in superclasses from a subclass using the syntax for generalized method invocation. The syntax is

```
(object AS class_name).method(...);
```

In the following example the Initialize method defined for Manager is calling the Initialize method for Employee:

```
CREATE METHOD Initialize()
RETURNS NULL
FOR Manager
BEGIN
    (SELF AS Employee).Initialize();
    ... -- more initialization here
END;
```

Enhancements

- LIKE predicate in a WHERE clause, e.g., LastName LIKE 'M%', can be optimized using an index.
- The Matisse SQL optimizer has been improved to speed up various query plans.
- The built-in function COUNT() used with a relationship, which returns the number of successor objects in a relationship, has been improved to execute faster.
- Arithmetic operations have been added for types DATE, TIMESTAMP, and INTERVAL

New C-API MtSQLParse

A new SQL C-API MtSQLParse() has been added, which parses an SQL statement and returns the type of the statement without execution.

These features are described in the *Matisse SQL Programmer's Guide*.

MOD built-in function

A built-in function MOD has been added, which returns the remainder of the first parameter divided by the second parameter.

1.5 .NET Binding

You can now assign `null` to an object relationship to clear all the successor objects in the relationship.

```
Manager mgr;  
.....  
mgr.Team = null;
```

1.6 Java Binding

A Selection named `DefaultSelection` is automatically generated when an object or a list of objects are returned by an SQL method (stored procedure) or a block statement. It can be used to generate a SQL projection or to enable cursored access.

2 Compatibility with Previous Releases

2.1 Data Migration

Matisse Server 7.0 comes with several changes in the data format for index optimization and unicode support. You must use the `mt_xml` tool for converting an existing database (6.x or prior) into the 7.0 format.

Step 1

Before installing 7.0.x, save your schema in ODL and your data in XML format:

```
mt_odl -d <dbname> -o schema.odl
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 7.0.x on your computer and then restore the schema and the data as follows:

```
mt_sd1 -d <dbname> import -odl schema.odl
mt_xml -d <dbname> data.xml
```

Note that the `mt_odl` functionality is now integrated in the new `mt_sd1` tool.

2.2 Client Connections

Only 7.0.9 clients may be used with 7.0.9 servers.

The clients for earlier releases of Matisse are incompatible with the 7.0.9 server. Consequently, you must upgrade any older clients to 7.0.9 before attempting to access a server running 7.0.9.

2.3 Java Binding

In the release 7.0.4, the code generator for the Java binding generates different stub classes for Date and Timestamp as well as for accessing schema objects.

If you are using the Java binding with the stub classes generated by the prior versions, you need to regenerate the stub classes using the `mt_sd1` utility.

```
> mt_sd1 stubgen -java schema.odl
```

For more information about code generation, refer to the *Matisse Java Programmer's Guide*.

3 Platform-Specific Topics

3.1 Solaris

Solaris 64-bit Kernel and Solaris on Intel

Installation packages for Solaris SPARC with 64-bit kernel and Solaris on Intel are available upon request.

3.2 FreeBSD

The FreeBSD version of Matisse is available upon request.

3.3 Linux

To be able to run the Matisse server on RedHat9, you need to use the pthread library located in `/lib/i686`, instead of the default library. This is enforced by setting the following variable in your initialization file (as `.bash_profile` or `.cshrc` in your home directory), for instance if you use `csh`:

```
setenv LD_ASSUME_KERNEL 2.4.1
```

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 7.0.9

- In some concurrent access cases, the MATISSE-E-WRITEWAITTIME error is returned before completion of the delay set by the user.
- MATISSE-E-SYSTEMERROR could be returned at commit time when large number of objects with variable size key indexes were deleted in one transaction.
- In SQL, the default memory quota has been increased from 10M to 500M eliminating the MATISSE-E-SQLSTACKOVF error when manipulating a few millions of objects in a single transaction.
- Added a new SQL statement: "SET MEMORY_QUOTA x" where x represents the maximum size of the server-side SQL workspace for a given connection. It is expressed in Mega Bytes. The default value is set to 500M. The minimum value is 50M. Note that setting a large value may cause a performance degradation if your server has insufficient memory.
- In SQL PSM, Stored Methods returning a list of objects could in some cases re-order by OID the object list instead of keeping the order defined by the user.
- Added "--oldpasswd" option to the "mt_user passwd" command so that it can be used in non-interactive mode.
- In Java the method getBytes defined on MtObject did not return a correct value. Generated methods for persistent classes with attributes of type BYTES are also affected by this issue.

Resolved in Matisse 7.0.8

- Restoring backups could fail with one of the 2 following errors:
GOM-F-NOENVROOTOBJ, Database root object not found – must init or restore
GOM-E-OBJNOTFOUND, Object <OID> not found

Resolved in Matisse 7.0.7

- Access to large relationships could fail with an internal Matisse Error (MATISSE-E-INTERNALERROR, Wrong data tag read).

Resolved in Matisse 7.0.6

- The .NET stub generator (mt_stbgen) does not generate correct data class definition when the schema class does not include any attribute or relationship.
- The IDataRecord.GetBytes method of the ADO.NET provider does not work properly when retrieving byte[] values.
- JDBC does not return blob type values.
- Stereotypes in Rational Rose UML are not handled properly.

- More scalar types have been supported by the Matisse Rose Link.
- The SQL DROP CLASS statement fails when the class contains self-referencing relationship
- The SQL compilation fails when a query statement has complex aggregation expression with relationship navigation and built-in function.
- OID comparison in SQL query has been enhanced.

Resolved in Matisse 7.0.5

- Within the Enterprise Manager's query analyzer, SQL statements containing ‘{‘ in a literal constant string do not execute because of the JDBC escape processing.
- Memory leaks in the Matisse client when executing SQL statements in some cases.
- SQL SELECT INTO statement may not work properly when used repeatedly within a stored method.
- The mt_odl utility does not work for a global relationship defined between classes.
- The mt_xml utility does not work properly when importing an XML document that misses values for the XML attribute ‘prealloc’ within processing instructions.
- Establishment of relationships in the Data Transformation Services does not work properly for some attribute types.
- Concurrent request of establishing database connections could fail when the number of requests is large because of operating system's limitation.
- Eiffel binding: {MT_LINKED_LIST}.wipe_out raises an error when the list is empty

Resolved in Matisse 7.0.4

- Missing MOD(Dividend, Divisor) built-in to calculate modulo in SQL.
- On Windows, the Enterprise Manager doesn't start when the current user account is different from the user account at installation.
- “Import Data ..” doesn't report errors properly.
- “Establish Relationships” doesn't work when PK/FK are defined on fixed size data types.
- Some SQL queries do not display the result correctly when executed from the Enterprise Manager.
- SQL built-in function CONCAT does not return a correct result when a parameter and the result variable are identical and the function is executed repeatedly in SQL Methods.
- The mt_stbgen utility for the .NET binding does not return error messages properly.

Resolved in Matisse 7.0.3

- The right click menu on the monitor window in the Enterprise Manager does not freeze the refresh, thus making it difficult to select a single line, for instance to abort a transaction.

- Several templates in the Enterprise Manager SQL Analyzer window are inaccurate or contain syntax errors.
- The `mt_stbgen` tool generates extra closing curly braces in some cases, when re-generating the stub classes.
- ODL export is including the Meta-Schema, both from Enterprise Manager and from the `mt_sdl` tool.
- Delete statements in stored methods in some cases ignores proper relationships of sub-classes involved in the deletion, thus leaving unbalanced inverse relationships.
- The `SELECTION` expression cannot be used in the `FROM` clause of a `SELECT` statement. For instance: `“SELECT * FROM SELECTION(Se11 UNION Se12) AS Class WHERE ...”`.

Resolved in Matisse 7.0.2

- `'mt_version -n'` removes all versions instead of only the one specified.
- Removing a Datafile may block and not be able to complete in some cases.
- Java binding date and timestamp cannot exceed year 2038.
- Empty strings not indexed in entry-point dictionaries.
- Spaces in Datafile path not supported.
- Concurrent update and delete of objects causes deadlock even in 'access for update' mode.
- The .NET code generation utility `mt_stbgen` may not generate correct code for SQL methods when there are overridden methods.
- `SQL DROP CLASS` statement does not delete related indexes and entry point dictionaries.
- SQL built-in function `COUNT` used with a relationship, which returns the number of successor objects in the relationship, returns `NULL` instead of `0` when the relationship is empty.
- Java binding may not properly release internal resources after disconnecting from a database.

Resolved in Matisse 6.1.1

- Date and timestamp in the Java binding do not support full range of available values.
- `SQL SELECT REF()` may not work in some specific environment.

Resolved in Matisse 6.1.0

- XML import truncates large attributes.
- ADO.NET interface `IDbConnection.ConnectionTimeout` returns `0`. It returns `1` in the new release.
- The .NET binding may not free external resources for database connection when garbage collector calls the `Dispose` method.

- Cannot obtain the class in the FROM clause from MtSQLGetStmtInfo.
- SQL syntax doesn't allow LIST (INT) instead of LIST (INTEGER).
- Schema import/export as odl in editor doesn't work with empty spaces in path.
- Database deadlocks during SQL execution may cause memory leaks on server side.
- MtSQLSetCurrent does not position on the relevant row in some cases, this also affects ODBC access.
- Some datatypes have errors when updating with ODBC SQLSetPos.
- User license key checks does not work properly for some hardware configurations.
- Large transactions with entry-point dictionary maintenance may cause an RPC error.
- With the Eiffel binding, putting an object in a specific place in MT_LINKED_LIST causes an error on post condition.
- When deleting objects with the Eiffel binding, some class invariants do not work correctly.
- The Eiffel binding may not free external resources correctly when garbage collector calls the Dispose method.

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 (FreeBSD available by request)
- *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 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.