

Matisse[®] 8.2.2

Release Notes

May 2009



Matisse 8.2.2 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 17 May 2009

Contents

| | | |
|----------|---|-----------|
| 1 | New Features in Matisse 8.2 | 6 |
| 1.1 | Overview | 6 |
| 1.2 | Matisse Lite | 6 |
| 1.3 | Enterprise Manager Tool | 7 |
| | Schema Viewer | 7 |
| | SQL Analyzer | 7 |
| | Code Generation | 7 |
| 1.4 | Matisse Data Transformation Services | 7 |
| | Data Transformation Services | 7 |
| | Many-To-Many With Intermediate File | 7 |
| | One-To-Many Ordered With Intermediate File | 8 |
| | Importing Composed Objects from a single File | 8 |
| | Exporting Composed Objects into a single File | 9 |
| 1.5 | Matisse Event Notification | 9 |
| 1.6 | Database Configuration Parameters | 9 |
| | DATEXTENDSIZ | 10 |
| | DATFULLINIT | 10 |
| | DATINITSIZ | 10 |
| | MAXSRVLOGFILES | 10 |
| | MAXBKPLOGFILES | 10 |
| 1.7 | License Key Format | 10 |
| 2 | New Features in Matisse 8.1 | 11 |
| 2.1 | Overview | 11 |
| 2.2 | Matisse SQL | 11 |
| | Renaming Schema Objects | 11 |
| | Managing Connection Options | 12 |
| | Dealing with Default Values | 13 |
| | Debugging SQL Methods | 14 |
| 2.3 | Enterprise Manager Tool | 15 |
| | Object Browser | 15 |
| | SQL Analyzer | 15 |
| 2.4 | Matisse XML Manager | 15 |
| | mt_xml | 15 |
| 2.5 | Matisse Data Transformation Services | 17 |
| | Data Transformation Services | 17 |
| | Incremental updates | 17 |
| | Many-To-Many New Option Tags | 17 |
| | Preserving the order in One-To-Many | 18 |
| | Multimedia Data | 18 |
| | Large Binary and Text Data | 19 |

| | | |
|----------|--|-----------|
| | Converting lists | 19 |
| | mt_dts | 20 |
| 2.6 | Java Binding | 21 |
| | Java 6 | 21 |
| 3 | New Features in Matisse 8.0 | 22 |
| 3.1 | Overview | 22 |
| 3.2 | Enterprise Manager Tool | 22 |
| | Remote Administration | 22 |
| | Server Operation Control | 22 |
| | Server Monitoring | 22 |
| | Database Monitoring | 23 |
| | Object Browser | 23 |
| | Backup/Restore | 23 |
| 3.3 | Server Manager Listener | 23 |
| 3.4 | Database Utility Commands | 23 |
| | mt_connection | 23 |
| | mt_server extendcache | 24 |
| | mt_server info | 24 |
| | mt_server setlicense | 24 |
| | mt_server checklicense | 24 |
| 3.5 | Database Configuration Parameters | 25 |
| | NAME | 25 |
| | AUTORESTART | 25 |
| 3.6 | Matisse SQL | 25 |
| | Returning a Table in a SQL Method | 25 |
| 3.7 | .NET Binding | 25 |
| | .NET 2.0 Support | 25 |
| | Namespaces Renaming | 25 |
| | Class Generator Tool | 25 |
| 3.8 | Java Binding | 26 |
| | Java 2 Platform 5.0 | 26 |
| 3.9 | XML Tool | 26 |
| | mt_xml | 26 |
| 3.10 | Open Source Interfaces and Language Bindings | 28 |
| 4 | Compatibility with Previous Releases | 29 |
| 4.1 | Matisse 8.2 Data Migration | 29 |
| | Step 1 | 29 |
| | Step 2 | 29 |
| 4.2 | Matisse 8.1 Data Migration | 29 |
| | Step 1 | 29 |
| | Step 2 | 29 |
| 4.3 | Matisse 8.0 Data Migration | 30 |
| | Step 1 | 30 |
| | Step 2 | 30 |

| | | |
|----------|---|-----------|
| 4.4 | Client Connections | 30 |
| 4.5 | License Key Format | 30 |
| | Release 8.0 | 30 |
| | Release 8.2 | 30 |
| 5 | Platform-Specific Topics | 31 |
| 5.1 | Linux | 31 |
| 5.2 | MacOS | 31 |
| 5.3 | Solaris | 31 |
| 5.4 | Windows | 31 |
| 6 | Update History | 32 |
| | Resolved in Matisse 8.2.2 | 32 |
| | Resolved in Matisse 8.2.1 | 33 |
| | Resolved in Matisse 8.2.0 | 35 |
| | Resolved in Matisse 8.1.4 | 36 |
| | Resolved in Matisse 8.1.3 | 37 |
| | Resolved in Matisse 8.1.2 | 38 |
| | Resolved in Matisse 8.1.1 | 40 |
| | Resolved in Matisse 8.1.0 | 42 |
| | Resolved in Matisse 8.0.6 | 43 |
| | Resolved in Matisse 8.0.5 | 43 |
| | Resolved in Matisse 8.0.4 | 44 |
| | Resolved in Matisse 8.0.3 | 44 |
| | Resolved in Matisse 8.0.2 | 45 |
| 7 | Documentation | 46 |
| | Matisse documents available on the Web | 46 |
| | Documents included with Matisse standard installation | 46 |
| | Open source bindings | 46 |

1 New Features in Matisse 8.2

1.1 Overview

The Matisse 8.2 release introduces new features and major enhancements in the Matisse product line:

- *Matisse Lite* is the embedded version of Matisse DBMS. Matisse Lite is a compact library that implements the server-less version of Matisse. Matisse Lite provides a transactional, multi-user database engine self-contained in an application process.
- The *Matisse Enterprise Manager* has been enhanced to improve the developers experience. The Schema Viewer has been extended to show detailed representation of Matisse schema objects. The SQL analyzer tool now provides timing information for each SQL statement execution. You can also now generate the source code corresponding the database schema classes within the Matisse Enterprise Manager.
- Matisse DTS has been enhanced to improve the loading time of very large many-to-many relationships into a Matisse database.
- Matisse Event Notification has been added to most Matisse language bindings.
- New optional database parameters have been added to improve the overall management of a Matisse database.

The Matisse 8.2 release also includes many improvements in SQL, language bindings, and the Matisse client library.

1.2 Matisse Lite

Matisse Lite is the embedded version of Matisse DBMS. Matisse Lite is a compact library that implements the server-less version of Matisse. Matisse Lite provides a transactional, multi-user database engine self-contained in an application process.

Matisse Lite shares with Matisse the same set of APIs thus eliminating the developer's learning curve. It also shares the same datafile format, making your application fully compatible in both environments.

An existing Matisse application can be converted to a Matisse Lite application by simply re-linking the application with the corresponding Matisse Lite Library.

Matisse Lite is ideally suited for applications which require zero configuration, no administration and a small footprint.

1.3 Enterprise Manager Tool

The *Matisse Enterprise Manager* has been enhanced to improve the developers experience. The Schema Viewer has been extended to show a detailed representation of Matisse schema objects. The SQL analyzer tool now provides timing information for each SQL statement execution. You can also now generate the source code corresponding the database schema classes within the Matisse Enterprise Manager.

Schema Viewer

A new tab has been added to the Schema Object Viewers to provide a detailed description of each schema object. The *Properties* tab displays a comprehensive description of a Matisse schema object. The *Advanced* tab has been added to show a detailed representation of a Matisse schema object.

SQL Analyzer

The SQL analyzer tool now provides timing information of the various query execution phases. The elapsed time for the compilation, execution and projection is displayed in the *Messages* tab window.

Code Generation

You can now generate the C#, VB.NET, Java, C++ and Eiffel source code corresponding the database schema classes within the Matisse Enterprise Manager by selecting the new “Generate Code” menu item associated to a Schema Node.

1.4 Matisse Data Transformation Services

Data Transformation Services

Matisse DTS has been enhanced to improve the loading time of very large many-to-many relationships into a Matisse database.

Many-To-Many With Intermediate File

A new feature allows you to create a many-to-many relationship from data stored in a CSV file. This new option removes the need to create and load the temporary intermediate table previously needed to create a many-to-many relationship. The use of a CSV file is described in XML with three new tags: *IntermediateFile*, *FileName* and *IntermediateColumn*. A many-to-many relationship descriptor with a CSV file is defined as follows:

```
<DTSlinks>
  <ManyToMany>
    <PrimaryKey Name="Member" Class="MemberProfile"
      Relationship="Friends">MemberId</PrimaryKey>
    <PrimaryKey Name="Friends" Class="MemberProfile"
      Relationship="Friends">MemberId</PrimaryKey>
    <IntermediateFile>
      <FileName>MemberFriends.csv</FileName>
      <IntermediateColumn
        AssociatedName="Member">MemberId</IntermediateColumn>
      <IntermediateColumn
        AssociatedName="Friends">FriendId</IntermediateColumn>
```

```

    </IntermediateFile>
  </ManyToMany>
</DTSlinks>

```

One-To-Many Ordered With Intermediate File

You can also create an ordered one-to-many relationship from data stored in a CSV file. The use of a CSV file is described in XML with three new tags: `IntermediateFile`, `FileName` and `IntermediateColumn`. An ordered one-to-many relationship descriptor with a CSV file is defined as follows:

```

<DTSlinks>
  <OneToMany Name="BookAuthor" PreserveOrder="True">
    <PrimaryKey Name="PersonId" Class="Author"
      Relationship="RecentPublications">PersonId</PrimaryKey>
    <PrimaryKey Name="ISBN" Class="Book">ISBN10</PrimaryKey>
    <IntermediateFile>
      <FileName>AuthorBooks.csv</FileName>
      <IntermediateColumn
        AssociatedName="PersonId">PersonId</IntermediateColumn>
      <IntermediateColumn
        AssociatedName="ISBN">ISBN10</IntermediateColumn>
    </IntermediateFile>
  </OneToMany>
</DTSlinks>

```

Importing Composed Objects from a single File

This new feature allows you to more easily import composed objects from a single CSV file. Composed objects are objects described with “part-of” relationships of type Composition.

The first line in the CSV file must list the fields name. Each field of the object parts is described by its full property path name. For example assuming the `Order` class described as follows:

```

interface Order : persistent
{
  attribute Integer OrderID;

  relationship PostalAddress BillAddress;
  relationship PostalAddress ShipAddress;
};

interface PostalAddress : persistent
{
  attribute String<16> Nullable City;
  attribute String<16> PostalCode;
};

```

The path to reach the Postal Code in the Billing Address is as follows:

```
BillAddress.PostalCode
```

The CSV file may look like the following:

```

$ more orders.csv
OrderID,BillAddress.City,BillAddress.PostalCode,ShipAddress.City,ShipAddress.PostalCode
10248,Bern,3012,Geneve,1204

```

The following command is importing composed objects in the `Order` class:

```
$ mt_dts -d example import orders.csv -c Order
```

Exporting Composed Objects into a Single File

Exporting composed objects from the `Order` class just require to execute a navigational SQL query such as:

```
SELECT OrderID,BillAddress.City AS
    "BillAddress.City",BillAddress.PostalCode AS
    "BillAddress.PostalCode",ShipAddress.City AS
    "ShipAddress.City",ShipAddress.PostalCode AS
    "ShipAddress.PostalCode" FROM "Order"
```

The following command is exporting all the composed objects from the `Order` class:

```
$ mt_dts -d example export orders.csv -sql "SELECT
    OrderID,BillAddress.City AS
    \"BillAddress.City\",BillAddress.PostalCode AS
    \"BillAddress.PostalCode\",ShipAddress.City AS
    \"ShipAddress.City\",ShipAddress.PostalCode AS
    \"ShipAddress.PostalCode\" FROM \"Order\""
```

The CSV file produced is as the following:

```
$ more orders.csv
"OrderID","BillAddress.City","BillAddress.PostalCode","ShipAddress.
    City","ShipAddress.PostalCode"
10248,"Bern","3012","Geneve","1204"
```

1.5 Matisse Event Notification

Matisse Event Notification is an efficient notification mechanism that allows developers to easily implement event-driven applications. In a distributed data processing environment, this notification mechanism allows one application or service to notify one or more application or service subscribers of any particular event. Matisse defines 32 user-events that can be manipulated individually or combined together and provides four functions `subscribe`, `unsubscribe`, `notify` and `wait` to manipulate these events.

Matisse Event Notification is now supported in most language bindings. The .NET, Java and C++ binding examples include a new project which demonstrates Matisse Event Notification mechanism.

1.6 Database Configuration Parameters

New optional database parameters have been added to improve the overall management of a Matisse database.

DATEXTENDSZ This parameter defines the minimum datafile extension size used when the datafile is full. This parameter is expressed in Megabytes. The minimum value of `DATEXTENDSZ` is 4 Megabytes. The maximum size is 128 Megabytes. The default value of the `DATEXTENDSZ` parameter is 10 Megabytes.

DATFULLINIT This parameter controls the datafile initialization size. When this parameter value is set to 1, the Matisse Server performs the full datafile initialization before becoming online. When it is set to 0, the Matisse Server is online as soon as the minimum datafile size (`DATINITSZ`) is initialized. The default value of the `DATFULLINIT` parameter is 0.

DATINITSZ This parameter defines the minimum database size to be initialized before the server is online. This parameter is expressed in Megabytes. The minimum value of `DATINITSZ` is 20 Megabytes. The maximum size is 256 Megabytes. The server uses the `DATINITSZ` parameter at initialization. The default value of the `DATINITSZ` parameter is 20 Megabytes.

MAXSRVLOGFILES This parameter defines the maximum version number of recycled server log files of the Matisse Server saved. The minimum value of `MAXSRVLOGFILES` is 1. The maximum value is 32. The default value of the `MAXSRVLOGFILES` parameter, which was 5 in prior releases, has been increased to 7.

MAXBKPLOGFILES This parameter defines the maximum version number of recycled backup log files of the Matisse Server saved. The minimum value of `MAXBKPLOGFILES` is 1. The maximum value is 32. The default value of the `MAXBKPLOGFILES` parameter, which was 5 in prior releases, has been increased to 7.

1.7 License Key Format

The customer license key format has changed in release 8.2. Matisse 8.2 does not recognize license keys issued for prior releases. Upon installation of Matisse 8.2, a license key with limited features is automatically issued.

2 New Features in Matisse 8.1

2.1 Overview

The Matisse 8.1 release introduces new features and major enhancements in the Matisse product line:

- *Matisse SQL* adds new DDL and DML syntax which are extensions of the SQL standard and introduces new features to help debug SQL methods.
- *Matisse XML Manager* now supports a new set of options to better manage very large XML files and provides performance improvements for both export and import of XML documents.
- The *Matisse Data Transformation Services* have been extended to ease the migration process for converting relational data into a Matisse data model. Matisse DTS now offers a rich option set for the extraction and loading of multi-media data. The performance for establishing millions of links between objects on large data sets has also improved.
- The *Java binding* now supports Sun's Java 6.
- Support for Windows Vista.

The Matisse 8.1 release also includes many improvements in SQL, language bindings, and the Matisse client library.

2.2 Matisse SQL

Renaming Schema Objects

The ALTER CLASS command has been extended to enable the renaming of an existing class, attribute or relationship.

```
Syntax  ALTER {CLASS | TABLE} class
        RENAME { TO new_class_name
                | ATTRIBUTE attribute TO new_attribute_name
                | RELATIONSHIP relationship TO new_relationship_name
        }
```

```
Example ALTER CLASS movie
        RENAME TO new_movies;

ALTER CLASS movie
        RENAME ATTRIBUTE category TO new_category;
```

Managing Connection Options

Connection options affect the way you can interact with the database. These options allow you to specify the type of access to the database, the object locking policy as well as the amount of time the server waits for access conflicts to be resolved.

Syntax

```
SET CONNECTION_OPTION DATA_ACCESS_MODE { DEFAULT
                                         | DATA_READONLY
                                         | DATA_MODIFICATION
                                         | DATA_DEFINITION
                                         }
```

Example

```
SET CONNECTION_OPTION DATA_ACCESS_MODE DATA_DEFINITION;
UPDATE schema objects ...;
COMMIT;
```

This option allows you to specify the type of access that you intend to use when connecting to the database. Possible values are:

- **DATA_READONLY** allows read only access to the data objects and to the schema. Any attempt to start a transaction will fail (only SET TRANSACTION READ ONLY is allowed).
- **DATA_MODIFICATION** (default) allows read/write access to the data objects and read only access to the schema.
- **DATA_DEFINITION** allows read/write access to the data objects and to the schema.

The first two access modes optimize the access to the schema. The **DATA_DEFINITION** access mode must be used only when schema or meta-schema updates are necessary.

This option cannot be changed when the transaction is in progress.

Syntax

```
SET CONNECTION_OPTION LOCKING_POLICY { DEFAULT
                                       | DEFAULT_ACCESS
                                       | ACCESS_FOR_UPDATE
                                       }
```

Example

```
SET CONNECTION_OPTION LOCKING_POLICY ACCESS_FOR_UPDATE;
```

This option allows the server to be configured to handle requests for read locks using write locks instead. The possible values are:

- **DEFAULT_ACCESS** (default): Normal behavior, requests for read locks result in read locks.

- `ACCESS_FOR_UPDATE`: Requests for read locks result in write locks.

This option may be changed at any time.

Syntax

```
SET CONNECTION_OPTION LOCK_WAIT_TIME { DEFAULT
                                        | NO_WAIT
                                        | WAIT_FOREVER
                                        | number of ms
                                        }
```

Example

```
SET CONNECTION_OPTION LOCK_WAIT_TIME 500;
```

This option allows you to specify the amount of time the server waits for access conflicts to be resolved; if a time-out occurs (wait-time expires), the explicit or implicit lock request is rejected. The possible values are:

- `NO_WAIT`: If the lock cannot immediately be granted, the lock request is released and the function returns immediately.
- `WAIT_FOREVER` (default): The server waits until there is a deadlock or until the lock is granted.
- A positive integer: This is the time (in milliseconds) that the server waits for the lock to be granted. If the wait-time expires, the lock request is rejected. If a deadlock occurs, the transaction fails or the lock request is rejected.

When multiple objects are requested, the wait-time applies to each object request individually. The wait-time affects the process of obtaining locks for reads and writes within transactions. Object version requests are affected neither by locks nor by wait-times.

Dealing with Default Values

An extension to the SQL standard has been added to the `UPDATE` command allowing to reset an attribute to its default value.

Syntax

```
UPDATE class SET
    attribute = { expression | DEFAULT | NULL } [, ...]
    relationship = expression [, ...]
    [WHERE search_condition]
```

Example

```
UPDATE movie SET rating = DEFAULT
    WHERE title = 'Thirteen Days';
```

Another extension to the SQL standard has been added to the `WHERE` clause allowing to check if an attribute has a default value.

The syntax for evaluation of a default value is as follows:

Syntax

```
expression IS [NOT] DEFAULT
```

Example `SELECT Title FROM movie WHERE rating IS NOT DEFAULT;`

Debugging SQL Methods

The new PSM_OUTPUT module allows developers to easily trace the execution of stored methods. The functions in this module enable you to print out variable name, type and content as well as messages from SQL methods into the database log file. The PRINT function prints out an expression value. The PRINT_LINE function prints out an expression value and then terminates the line. The PRINT_VARIABLE function prints out detailed information about a variable (name, type and value) and then terminates the line. The ENABLE function enables calls to PRINT, PRINT_LINE and PRINT_VARIABLE. Calls to these functions are ignored if the PSM_OUTPUT module is not enabled. The DISABLE function disables calls to PRINT, PRINT_LINE and PRINT_VARIABLE, and purges the message buffer of any remaining information.

Syntax

```
PSM_OUTPUT.PRINT(<expression>)
PSM_OUTPUT.PRINT_LINE(<expression>)
PSM_OUTPUT.PRINT_VARIABLE(<expression>)
PSM_OUTPUT.ENABLE()
PSM_OUTPUT.DISABLE()
```

Example

```
BEGIN
  DECLARE Obj Person;
  [...]
  PSM_OUTPUT.PRINT(Obj.LastName);
  PSM_OUTPUT.PRINT_LINE(' ');
  PSM_OUTPUT.PRINT_LINE(CONCAT('printTrace() - ',Obj.FirstName));
  PSM_OUTPUT.PRINT_VARIABLE(obj.OID);
END
```

Excerpt from the log file after running the SQL statement:

```
24 Oct. 2007 18:03:58 PSM Output: Washington
24 Oct. 2007 18:03:58 PSM Output: printTrace() - Georges
24 Oct. 2007 18:03:58 PSM Output: V2 (MT_OID) = 0x1086
```

The following example disables calls to PRINT, PRINT_LINE and PRINT_VARIABLE, and purges the message buffer of any remaining information:

```
BEGIN
  PSM_OUTPUT.DISABLE();
END
```

The example below enables calls to PRINT, PRINT_LINE and PRINT_VARIABLE:

```
BEGIN
```

```

        PSM_OUTPUT.ENABLE ();
    END

```

2.3 Enterprise Manager Tool

The *Matisse Enterprise Manager* has been enhanced to improve the developers experience. The Object Browser has been extended to ease the manipulation of object hierarchies. The SQL analyzer tool provides a new SQL syntax highlighting editor.

Object Browser Two new tabs have been added to the Object Browser to provide an easy access to object hierarchies. The *Advanced* tab has been added to make both standard and advanced query builders more independent of each other and easier to use. The *Statistics* tab presents in a table format the object count for each class in the database. A right-click on a class in the table allows the user to view some objects from the selected class and its sub-classes.

SQL Analyzer The SQL analyzer tool now provides a new SQL syntax highlighting editor to improve the readability of SQL statements and methods. Combined with a large set of predefined SQL templates, this new feature enables you to write correct SQL query statements and SQL methods faster.

2.4 Matisse XML Manager

Matisse XML Manager has been enhanced to improve the management of very large XML files. The `-s` option added to the export command allows the users to split the database export into multiple XML files. Media data is also exported into external files by default. To export media data into the XML document, you now need to add the `-emedia` option to the export command. By default the import operation is now loading the database in multiple transactions (20K objects per transaction). The `-commit` option allows the user to control the volume of objects created in each transaction. The `-scommit` added to the import command forces to import the XML document into a single transaction.

mt_xml The command line options of Matisse XML import/export utility are as follows:

```

$ mt_xml
MATISSE XML Manager x32 Version 8.1.0.0 (32-bit Edition) - Oct 12
  2007.
(c) Copyright 1992-2007 Matisse Software Inc. All rights reserved.

Usage:
  Import:
    mt_xml [-v] -d [<user>:]<database>[@<host>[:<port>]] [-p]
            import {-f <xml_file> | -in} [-update] [-commit <n> | -
            scommit]

```

```

Export:
  mt_xml [-v] -d [<user>:]<database>[@<host>[:<port>]] [-p]
        export {-f <xml_file> [-s <size>] | -out} [-foid] [-emedial]
        {-full | -sql <stmt> | -oid <oid>...}

Parse:
  mt_xml parse {-f <xml_file> | -in}

-d [user:]database[@host]: Specifies the database to be accessed.

-p                               : Allows the user to authenticate with a
  specific                        username/password.
                                - if the '-p' option is used, it will
                                be assumed that the current system user
                                is known from the database, but a
                                password will be asked.
                                - if the '-p' option is not used, Matisse
                                <user> is used for user name, and a
                                password will be asked.
                                - if the user is not defined and
                                the '-p' option is not used, it will
                                be assumed that the current system user
                                is known from the database, and does not
                                need password.

-v                               : Reports the number of objects
  imported/exported.

import -f <xml_file> : Reads the XML file and imports the XML data into the database.
                    Arguments appearing within {} indicate that one
                    of the arguments must be specified.

import -in           : Reads the XML data from standard input and
                    import it into the database.

-update             : When specified with MtPrimaryKey attribute,
                    values of existing objects are updated.

-commit <n>         : Commits transaction for every <n> objects
                    created. by default commit occurs every 20480 objects
                    created

-scommit           : Forces to import the XML document in a single
                    transaction
                    requires enough memory to parse the XML document
                    and to create all the objects in memory

export -f <xml_file> : Generate the XML file containing XML data
                    extracted from the database specified by
                    either a SQL statement <stmt> or by <oid>s.
                    Arguments appearing within {} indicate that one
                    of the arguments must be specified.

-s <size>          : Specifies the XML data file max size therefore
                    splitting XML data into multiple XML files named
                    <db name>_xds_<document id>.xml.
                    The file size is in Giga bytes

```

`export -out` : Writes XML data to the standard output.
`-full` : Exports all non-schema data into a single XML file.
`-sql <stmt>` : Specifies objects to be exported using a SQL statement `<stmt>`.
`-oid <oid>` : Specifies OIDs of objects exported. Both decimal and hexadecimal oid are accepted.
`-foid` : Exports data in a format with OIDs in the xml tags to enable Primary Key recovery. The `-full` option always exports in this format
`-emedia` : Exports media data in the XML document instead of exporting media data into external files.
`parse -f <xml_file>` : Parses the XML data file and indicates if the XML data format is valid or not
`parse -in` : Reads the XML data from standard input and parses it.

2.5 Matisse Data Transformation Services

Data Transformation Services

Matisse DTS has been enhanced to improve the loading time of large relational databases into a Matisse database. DTS has also been extended to ease the conversion of relational data into Matisse. DTS can now load incremental updates from a relational database that enable a rapid synchronization with a Matisse database used for example as fast caching in the middle-tier. Matisse DTS now offers a rich option set for the extraction and loading of multi-media data.

Incremental updates

The `-update` option of the `import` command enables the updates of existing objects when specified with the first columns of the CSV file composing a primary key. Then applying the `link` command incrementally updates the relationship between objects.

Many-To-Many New Option Tags

Two new option tags have been added to ease the description of many-to-many associations. The `Name` option defined on the `PrimaryKey` tag and the corresponding `AssociatedName` option defined on the `IntermediateKey` tag make the relationship description more compact and more easy to read. These options are particularly useful when the direct relationship and its inverse are identical. For example, the relationship `Friends` defined on the `Person` class is described as follows:

```
<DTSlinks>
  <ManyToMany>
```

```

<PrimaryKey Name="Member" Class="MemberProfile"
  Relationship="Friends">MemberId</PrimaryKey>
<PrimaryKey Name="Friends" Class="MemberProfile"
  Relationship="Friends">MemberId</PrimaryKey>
<IntermediateTable>
  <TableName>MemberFriends</TableName>
  <IntermediateKey
    AssociatedName="Member">MemberId</IntermediateKey>
  <IntermediateKey
    AssociatedName="Friends">FriendId</IntermediateKey>
</IntermediateTable>
</ManyToMany>
</DTSlinks>

```

Preserving the order in One-To-Many

A new link feature allows you to preserve the order of the elements in one-to-many association. To establish a one-to-many relationship which preserves order, you need to define an intermediate table that describes the association elements in the order they will be created. The relationship descriptor in XML needs to include the `PreserveOrder` tag as well as an intermediate table description as follows:

```

<OneToMany Name="ManagedProjectsRel" PreserveOrder="TRUE">
  <PrimaryKey Class="ProjectManager"
    Relationship="Manages"
    DeleteAfter="True">EmpId</PrimaryKey>
  <PrimaryKey Class="Project"
    Relationship="ManagedBy"
    DeleteAfter="True">ProjectId</PrimaryKey>
  <IntermediateTable DeleteAfter="True">
    <TableName>ManagedProjects</TableName>
    <IntermediateKey
      AssociatedClass="ProjectManager"
      AssociatedPrimaryKey="EmpId">EmpId</IntermediateKey>
    <IntermediateKey
      AssociatedClass="Project"
      AssociatedPrimaryKey="ProjectId">ProjectId</IntermediateKey>
  </IntermediateTable>
</OneToMany>

```

Multimedia Data

Matisse DTS allows you to import and to export multimedia data into files. The `ColumnFromFile` and `ColumnToFile` XML options tags have been added respectively to the XML Import and XML Export options descriptors.

Import Tag:

```
<ColumnFromFile Directory="/import/images">Photo</ColumnFromFile>
```

Export Tag

```
<ColumnToFile Directory="/export/images"
  FilenameFormat="img_{LastName}.jpg">Photo</ColumnToFile>
```

Using the `ColumnToFile` tag enables to customize the media filename to be exported.

```
<columnToFile Directory="C:\photos"
```

```
FilenameFormat="{MediaName}">Photo</columnToFile>
```

The ‘Directory’ tag can define the files location (i.e. Directory="C:\photos"). By default, if the ‘Directory’ tag is omitted, the media files are created in the same directory as the CSV file.

The ‘FilenameFormat’ tag defines the file name format. It can be composed of text and parameters. A parameter can be an column name, RowId, or OID. a Parameter name is defined inside curly braces ({}).

The ‘FilenameFormat’ tag defines the file name format. It can be composed of text and parameters. A parameter can be an column name, RowId, or OID. a Parameter name is defined inside curly braces ({}).

If the ‘FilenameFormat’ tag is omitted, the filename format is as follows:

```
{ClassName}_{ColumnName}_{RowId}.{data type}
```

This default format for attribute Photo of class Employee will generate filenames such as Employee_Photo_1.img, Employee_Photo_2.img, Employee_Photo_3.img, etc.

For example, exporting images using the OID parameter as defined below will produce media filename such as image_5334.jpg, image_5338.jpg, image_5359.jpg, etc.:

```
<DTSEXport>
  <selectStatement>SELECT c.MediaName,c.Photo FROM PhotoShot
  c</selectStatement>
  <columnToFile Directory="C:\Export\photos"
    FilenameFormat="image_{OID}.jpg">Photo</columnToFile>
</DTSEXport>
```

For example, exporting images using a column name as parameter as defined below will produce media filename equal to the value of MediaName:

```
<DTSEXport>
  <selectStatement>SELECT c.MediaName,c.Photo FROM PhotoShot
  c</selectStatement>
  <columnToFile Directory="C:\Export\photos"
    FilenameFormat="{MediaName}">Photo</columnToFile>
</DTSEXport>
```

Large Binary and Text Data

By default, large binary data (MT_BYTES) and large text data (MT_TEXT) are imported as field values. Using the ColumnFromFile and ColumnToFile option tags allow to manage the values into files as for media data.

Converting lists

Matisse DTS exports Matisse Lists as one list element per row.

```
"SSN", "Name", "Hobby", "E-mails"
1234, "John", "Bridge", "john@gmail.com"
1234, "John", "Painting", "john@yahoo.com"
```

```
1234, "John","Violin",
1234, "John","Soccer",
1234, "John","Golf",
```

Matisse DTS converts multi-row values into Matisse Lists. To avoid data duplication in your CSV files, we recommend that you use one CSV file for each column of type list. The example above with 2 attributes (Hobby and E-mails) of type `STRING_LIST` will slip in 3 CSV files as follows:

```
"SSN", "Name"
1234, "John"

"SSN", "Hobby"
1234, "Bridge"
1234, "Painting"
1234, "Violin"
1234, "Soccer"
1234, "Golf

"SSN", "E-mails"
1234, "john@gmail.com"
1234, "john@yahoo.com"
```

mt_dts

The command line options of Matisse DTS import/export utility are as follows:

```
$ mt_dts
MATISSE Data Transformation Services x32 Version 8.1.0.0 (32-bit Edition) - Oct 12 2007.
(c) Copyright 1992-2007 Matisse Software Inc. All rights reserved.
```

Usage:

Import:

```
mt_dts -d [<user>:]<database>[@<host>[:<port>]] [-p] [-o <options file>] import <CSV file> [-c
<class name>] [-update]
```

Export:

```
mt_dts -d [<user>:]<database>[@<host>[:<port>]] [-p] [-o <options file>] export <CSV file> [-sql
"<SQL select>" | -c <class name>]
```

Link:

```
mt_dts -d [<user>:]<database>[@<host>[:<port>]] [-p] [-o <options file>] link <XRD file>
```

-d [user:]database[@host] : the database to be accessed

-p : allows the user to authenticate with a username/password.
- if the '-p' option is used, it will be assumed that the current system user is known from the database, but a password will be asked.
- if the '-p' option is not used, Matisse <user> is used for user name, and a password will be asked.
- if the user is not defined and the '-p' option is not used, it will be assumed that the current system user is known from the database, and does not need password.

-o <options file> : The file containing the Import/Export options in XML format

import <CSV file> : The CSV file to be loaded

-c <class name> : The class name where the data will be loaded if different from the CSV filename

-update : When specified with the first columns of the CVS file composing the primary key, values of existing objects are updated.

export <CSV file> : The CSV file to be generated

-sql <SQL Select> : The SQL select statement which filters data to be exported, or

-c <class name> : The class containing the data to be exported

link <XRD file> : The XML Relationship Descriptors file describing how to establish relationship between entities

2.6 Java Binding

Java 6

Matisse Java binding has been upgraded to support the new Sun Java 6.

3 New Features in Matisse 8.0

3.1 Overview

The Matisse 8.0 release introduces major features in the Matisse product line:

- The *Matisse Enterprise Manager* now provides full remote administration features for distributed Matisse database servers on a local network.
- The *.NET binding* now supports Microsoft's .NET Framework 2.0 and Visual Studio 2005.
- The *Java binding* now supports Sun's Java 2 Platform 5.0.
- Matisse 64-bit is now available on Linux and Windows.
- Support for Solaris 10 on both SPARC and AMD Opteron chips.

The Matisse 8.0 release also includes many improvements in SQL, language bindings, and the Matisse client library.

3.2 Enterprise Manager Tool

The *Matisse Enterprise Manager* has been redesigned to regroup in a single tool: the distributed management of database servers, the management of database schemas, the data import and export in table (relational) and XML formats, as well as various security and administration functions. It includes an Object Viewer to browse and edit object hierarchies stored in a database. It also includes a SQL analyzer tool to help optimize complex queries and produce result-sets in a table format.

Remote Administration

The *Enterprise Manager* now provides full remote administration features for distributed Matisse database servers on a local network. All administration tasks can be executed remotely via the enterprise manager. This includes database start up and shutdown, database backup and restore, server monitoring, database monitoring, database access control and database schema and data manipulation.

Server Operation Control

The Server Operation Control component provides security control for executing local or remote administration operations including start/stop databases, backup/restore and datafile management.

Server Monitoring

The *Enterprise Manager* monitors registered database servers detecting in real-time when servers, databases and Matisse services become unavailable. It also provides real-time monitoring of CPU activity, memory consumption and disk usage of database servers.

Database Monitoring

The Database Monitoring component presents real-time detailed information on a selected database. It provides information on the active connections to the database, transactions performed on the database, and data storage activity. This tool replaces the former `mt_monitor` utility.

Object Browser

The new Object Browser allows users to access object hierarchies stored into a database. Matisse Standard and Advanced Query Builders provide an easy access to objects. You can then navigate through the object result-set and update object properties.

Backup/Restore

Matisse Database Backup allows users to perform full and incremental parallel backups of databases while the system is online. There is no need to block updates during a backup, as the backup sub-system uses a snapshot of the database at the time of the beginning of the backup operation.

Matisse Database Restore provides wizards to guide administrators through the restore process. When restoring, you must first start your database for restore, then you can start the restore process. For restoring from a multi-increment backup, you can restore the full backup files and the incremental backup files in any order, either sequentially or in parallel, and then shutdown and restart the database complete the process.

3.3 Server Manager Listener

Matisse Server Manager Listener (SMListener) manages remote operation requests on a local network. The `mt_smlistener` daemon also controls the denial of operations execution on the machine it is running on. The SMListener daemon is responsible for creating new databases, starting and stopping databases as well as managing backups and restore operations. The SMListener utility is also responsible for restarting database servers automatically after a reboot of the machine.

3.4 Database Utility Commands

`mt_connection`

The `mt_connection` utility allows you to view connections and kill active connections. A Connection ID field has been added to the connection description. The `--cid` option has been added to the `kill` command to designate a connection to be killed by its ID. Now connections can be killed individually in a multi-threaded environment with multiple connections.

```
$ mt_connection -d example list
2 connection(s):
ID  NODE  USER  PID  CALLS  TRANID  BLOCKED  FUNCTION
7426 JADE  john  20593  1      1      connections
7424 JADE  mary  20587  26     26     exec sql
```

mt_server extendcache

The `mt_server extendcache` command allows you to extend the size of the server cache on a running database.

```
$ mt_server -d example extendcache -s 1024M
Server cache size extended to 1024M
```

mt_server info

The `mt_server info` command, which provides information for a running database, now displays information in a format similar to the Enterprise Manager Tool.

```
$ mt_server -d example info
Database:
  Name: example
  Server version: 8.0.3
  Datafile version: 8.0.3
Status:
  Start date: 05 Jul. 2006 15:33:13
  Server uptime: 52 sec
  Backup date: not available
  Version collect date: not available
Configuration:
  size: 8 Kbytes
  cache size: 32 Mbytes
  Server failover: disabled
  Datafile extension: automatic
  Access control: disabled
  Version collection: automatic
Current State:
  Transaction manager: enabled
  Current logical time: 2
  Highest collected logical time: 1
  Last backup logical time: 0
```

mt_server setlicense

The `mt_server setlicense` command allows you to set the customer license key on your server. The `setlicense` command replaces the `license` command available in release 7.

mt_server checklicense

The `mt_server checklicense` command allows you to check the customer license key installed on your server. The `--full` option provides a complete description of the installed license.

```
$ mt_server checklicense
Your 31 days license expires in 15 days
```

```
$ mt_server checklicense --full
License Description:
Floating License - Standard Edition - up to 5 users - up to 4 logical CPUs
License options - mirroring disabled, raw partition datafiles disabled, replication disabled
License expires in 474 days
```

3.5 Database Configuration Parameters

- NAME** This parameter defines the name of the database. The maximum number of characters for a database name, which was 12 in prior releases, has been increased to 31 characters.
- AUTORESTART** This parameter establishes the automatic restart of a database when the host machine is rebooted. When this parameter value is set to 1, the Matisse Server Manager Listener restarts the database server automatically upon reboot. When it is set to 0, no action is performed upon reboot.

3.6 Matisse SQL

- Returning a Table in a SQL Method** A new syntax has been added to SQL methods for returning tables. For instance, the following SQL method returns a table listing the presidents in office between `startYear` and `endYear`:

```
CREATE STATIC METHOD viewPresidentsBetween(startYear INT, endYear INT)
RETURNS TABLE(firstName VARCHAR(32), lastName VARCHAR(32), startingYear INT,
                endingYear INT)
FOR Presidency
BEGIN
    SELECT p.FirstName, p.LastName, p.IsInChargeOf.StartingYear, p.IsInChargeOf.EndingYear
    FROM Person p
    WHERE p.IsInChargeOf.EndingYear >= startYear
    AND p.IsInChargeOf.StartingYear <= endYear;
END;
```

3.7 .NET Binding

- .NET 2.0 Support** Matisse .NET binding has been upgraded to support Microsoft's new .NET Framework 2.0 and Visual Studio 2005. In Release 8, the support for .NET Framework 1.1 has been discontinued.

- Namespaces Renaming** To align Matisse .NET binding to Microsoft .NET framework naming conventions, `com.matisse` namespace has been renamed `Matisse` and `com.matisse.reflect` namespace has been renamed `Matisse.Reflect`.

- Class Generator Tool** The Matisse .NET Object Manager utility `mt_dnom` generates C# or VB.NET source codes. This utility generates source codes for calling SQL methods in the database server without using a SQL statement, which seamlessly extends object-oriented programming to the database server. 'ADO Data Classes' can be also generated in addition to the regular stub class generation by the `mt_dnom` utility. The 'ADO Data Classes' are useful when you need to copy the Matisse property

values to your application classes that are independent from the persistence layer, for example serializing objects for transfer over the network. The `mt_dnom` utility now supports user authentication.

\$ `mt_dnom`

Matisse .NET Object Manager x32 Version 8.0.3 (32-bit Edition) - Sep 21 2006.

(c) Copyright 1992-2006 Matisse Software Inc. All rights reserved.

Usage:

Generate Stubs:

```
mt_dnom -d [user:]database[@host[:port]] [-p] stubgen [-lang C# | VB] [-n <namespace>] [-adc <namespace>] [-no]psm
```

`-d [user:]database[@host]` : the database to be accessed.

`-p` : allows the user to authenticate with a username/password.
- if the `-p` option is used, it will be assumed that the current system user is known from the database, but a password will be asked.
- if the `-p` option is not used, Matisse `<user>` is used for user name, and a password will be asked.
- if the user is not defined and the `-p` option is not used, it will be assumed that the current system user is known from the database, and does not need password.

```
stubgen [-lang C# | VB] [-n <namespace>] [-adc <namespace>] [-no]psm
```

`-lang` : generate C# or VB files from the database schema. The default is C#

`-n <namespace>` : define the generated classes in the specified namespace

`[-no]psm` : generate .NET methods mapping Persistent SQL methods. The default is `-psm`

`-adc <namespace>` : generate ADO Data Classes in addition to stub classes

The Matisse .NET Object Manager utility `mt_dnom` utility replaces the former `mt_stbgen` utility.

3.8 Java Binding

Java 2 Platform 5.0 Matisse Java binding has been upgraded to support the new Sun Java 2 Platform 5.0.

3.9 XML Tool

mt_xml The Matisse XML import/export utility has been updated to provide command line options similar to the other Matisse utilities.

\$ `mt_xml`

MATISSE XML Manager x32 Version 8.0.3 (32-bit Edition) - Sep 21 2006.
 (c) Copyright 1992-2006 Matisse Software Inc. All rights reserved.

Usage:

Import:

```
mt_xml [-v] -d [<user>:]<database>[@<host>[:<port>]] [-p]
import {-f <xml_file> | -in} [-update] [-commit <n>]
```

Export:

```
mt_xml [-v] -d [<user>:]<database>[@<host>[:<port>]] [-p]
export {-f <xml_file> | -out} [-foid] {-full | -sql <stmt> | -oid <oid>...}
```

-d [user:]database[@host]: Specifies the database to be accessed.

-p : Allows the user to authenticate with a specific username/password.
 - if the '-p' option is used, it will be assumed that the current system user is known from the database, but a password will be asked.
 - if the '-p' option is not used, Matisse <user> is used for user name, and a password will be asked.
 - if the user is not defined and the '-p' option is not used, it will be assumed that the current system user is known from the database, and does not need password.

-v : Reports the number of objects imported/exported.

import -f <xml_file> : Import the XML data into the database.
 The export option allows to generate this file containing XML data extracted from the database. Arguments appearing within {} indicate that one of the arguments must be specified.

import -in : Reads the XML data from standard input and import it into the database.

-update : When specified with MtPrimaryKey attribute, values of existing objects are updated.

-commit <n> : Commits transaction for every <n> objects parsed.

export -f <xml_file> : Generate the XML file containing XML data extracted from the database by specifying either a SQL statement <stmt> or by <oid>s.
 Arguments appearing within {} indicate that one of the arguments must be specified.

export -out : Writes XML data to the standard output.

-full : Exports all non-schema data into a single XML file.

- sql <stmt> : Specifies objects to be exported using a SQL statement <stmt>.
- oid <oid> : Specifies OIDs of objects exported. Both decimal and hexadecimal oid are accepted.
- foid : Exports data in a format with OIDs in the xml tags to enable Primary Key recovery. (The -full option always exports in this format)

3.10 Open Source Interfaces and Language Bindings

The following bindings are currently available in open source and have been updated for release 8:

- Eiffel binding 8
- Perl binding 8
- PHP binding 8
- Python binding 8
- Smalltalk binding 8

These bindings are all provided as open source. Developers are encouraged to contribute to these bindings under Matisse Interface Public License (MIPL), which is a variant of the Mozilla Public License.

You may check the download for the current status of the open source bindings:

<http://www.matisse.com/developers/downloads/>

4 Compatibility with Previous Releases

4.1 Matisse 8.2 Data Migration

Matisse Server 8.2 requires the conversion of an existing database running on a 8.1.x server. You must use the `mt_xml` tool for converting an existing database (8.1.x or prior) into the 8.2.x format.

Step 1

Before installing 8.2.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
```

Step 2

You may now install Matisse 8.2.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
```

4.2 Matisse 8.1 Data Migration

Matisse Server 8.1 requires the conversion of an existing database running on a 8.0.x server. You must use the `mt_xml` tool for converting an existing database (8.0.x or prior) into the 8.1.x format.

Step 1

Before installing 8.1.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
```

Step 2

You may now install Matisse 8.1.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
```

4.3 Matisse 8.0 Data Migration

Matisse Server 8.0 comes with several changes in the data format. You must use the `mt_xml` tool for converting an existing database (7.x or prior) into the 8.0 format.

Step 1

Before installing 8.0.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 8.0.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
```

4.4 Client Connections

Only 8.2.x clients may be used with 8.2.x servers.

The clients for earlier releases of Matisse are incompatible with the 8.2.x server. Consequently, you must upgrade any older clients to 8.2.x before attempting to access an 8.2.x server.

4.5 License Key Format

Release 8.0

The customer license key format has changed in release 8.0. Matisse 8.0 does not recognize license keys issued for prior releases. Upon installation of Matisse 8.0, a license key valid for 30 days is automatically issued.

Release 8.2

The customer license key format has changed in release 8.2. Matisse 8.2 does not recognize license keys issued for prior releases. Upon installation of Matisse 8.2, a license key with limited features is automatically issued.

5 Platform-Specific Topics

5.1 Linux

The following Linux distributions on x86 (32-bit) and AMD64 and EM64T (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.

5.2 MacOS

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

5.3 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.

5.4 Windows

Support for Windows (2000/XP/2003/Vista) on systems based on x86 (32-bit) and x86_64 (64-bit) chips families.

6 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 8.2.2

- In SQL, `SELECT DISTINCT` has been extended to support relationship navigation.
- In SQL, the `ORDER BY` clause has been extended to support relationship navigation.
- In SQL, the `GROUP BY` and `HAVING` clauses have been extended to support relationship navigation.
- In SQL, the server-side execution of the `REF()` built-in supports relationship navigation.
- In SQL, A SQL Method returning a `TABLE` can now returns column of Object types.
- In SQL, `SELECT UNFILTERED` used in a direct statement forces the projection to be built on the server-side the same way a block statement or a SQL Method would do it.
- In SQL, `SELECT FILTERED` applies the relationship navigation filters defined in the `WHERE` clause to the matching relationships used in the Select list. The SQL projection produced is equivalent to the projection of a SQL relational equi-join.
- 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.
- In Matisse 8.0.x adding a 0-to-1 relationship with the SQL DDL statement (`ALTER CLASS MyClass ADD RELATIONSHIP myRel REFERENCES LIST (MtSucClass) CARDINALITY (0,1) INVERSE MySucClass.myInvRel;`) did not require the `LIST` keyword to qualify the reference class to generate a relationship that is equivalent to relationship defined in ODL as follows:

```
MyClass myRel[0,1] inverse MySucClass::myInvRel;
```
- In SDL, Methods created on Meta-schema classes (i.e. `MtObject`, `MtClass`) are not exported in the ODL file.
- In the Matisse Java Binding, Matisse Object Iterators have been extended to support Java Generics.
- In the Matisse Java Binding, the generated code for persistent classes includes object iterators to enumerate the class own instances (excluding subclasses).
- In the Matisse Java Binding, the generated code for persistent classes now includes new checking methods for each attribute:

```
MyClass.is<MyAttribute>Null() (only for nullable attribute)
MyClass.is<MyAttribute>DefaultValue()
```
- The Java binding examples include new projects which demonstrates Connection Pooling, Class Reflection and SQL Methods in Java.
- The Matisse Java Binding manual now includes more specific examples about Connection Pooling, Database Events, Class Reflection, SQL Methods in Java, etc.

- In the Matisse Java Binding, the `MtObject.getTimestamp()` method returns a Null Pointer Exception when the attribute value is `MT_NULL` while `null` is expected.
- In Matisse JDBC, when the `MtCallableStatement.getObject()` method returns a selection of Matisse objects, the type of the objects array does not match the class of the selection, but is always of type `MtObject`. This issue prevents from casting the array into the expected object type such as:

```
Employee[] sel = (Employee[])stmt.getObject(0);
```
- Adding an Entry Point Dictionary on an attribute of type `Boolean` when the class has already millions of instances may fail.
- In DTS, updating objects with `-update` option does not return the expected error when the attribute in the first column in the CSV file has an index which not a primary key.
- On Windows, all the Matisse DLLs now include a File Version number synchronized with the Release Version.
- Added `monitor` command to the `mt_server` command which monitors 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
```
- Added `count` command to `mt_connection` which returns only 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.
- The restart time of large checkpointed database (> 100M objects) has been significantly improved.
- In Matisse Enterprise Manager SQL Analyzer on Linux, a Java repainting exception may occur when the result-set table is being displayed.

Resolved in Matisse 8.2.1

- In the Matisse .NET binding, the Matisse Stub Class generator has been extended to support any .NET programming language that includes a .NET CodeDOM provider. With this new feature, you can generate Matisse Stub Class to manipulate your persistent data from the .NET programming language of your choice. A new VS project has been added to illustrate this new feature.
- In the Matisse .NET binding, the Matisse Stub Class generator now generates new tags to ease the updates of existing source code files. It also creates a 'Matisse Generated Code' region in the programming languages that support this feature.

```
// GEN_START: Matisse Generated Code - Do not modify
#region Matisse Generated Code - Do not modify
// Generated with Matisse .NET Object Manager x32 Version 8.2.1

#endregion
```

```
// GEN_END: Matisse Generated Code - Do not modify
```

The new Matisse Stub Class generator does not update existing source code files. Therefore you need to generate new files from your database schema and re-apply by hand the changes, if any, you have made in your existing source code files.

- The Matisse .NET binding now includes a new VS project which demonstrates Matisse support for Time-Series in .NET.
- The Matisse .NET binding now includes a new series of examples for IronPython (Python for .NET) which demonstrates Matisse support for .NET programming languages that goes beyond Microsoft supported .NET programming languages.
- In the Matisse .NET binding, some of the resources allocated by `MtCommand` are not properly released by `Dispose()` causing the process memory to grow overtime.
- In Matisse Enterprise Manager SQL Analyzer Editor, an option to show or hide the source code line number has been added.
- In Matisse Enterprise Manager SQL Analyzer, the query results are now displayed page-by-page.
- In Matisse Enterprise Manager Object Browser, detailed timing for a query execution is now reported in the message window.
- In Matisse Enterprise Manager SQL Analyzer, in some cases the format for `Date` and `Timestamp` datatypes displayed in a table result set may be invalid. For example, the micro-seconds displayed may be inconsistent with the real value stored in the database.
- In Matisse Enterprise Manager Object Browser, the database schema displayed may be de-synchronized with the most recent one stored in the database until the `Refresh` menu-item on the `Database` node is activated. This situation occurs if the database server is stopped and then restarted with a new database schema outside of the running Matisse Enterprise Manager.
- Matisse Data Transformation Services (`mt_dts`) now support importing composed objects from a single CSV file. Composed objects are objects described with “part-of” relationships of type `Composition`.
- Matisse Data Transformation Services have been extended to support a less rigid `Date` and `Timestamp` datatypes format. For example, this new extension simplifies loading CSV data produced by “Copy&Paste” of rows from a `GRID` object in an application or in a development environment tool.
- When linking objects with the `mt_dts` utility if an `UNMATCHEDFKKEY` error occurs, the primary key value displayed in the error message may not be readable.
- The `mt_dts` utility may fail to link objects when the relationship is of type one-way relationship.
- In SQL PSM, passing `SQL INTEGER` variables to the `OFFSET` and `LIMIT` clauses causes the query compilation to fail with a syntax error.
- In SQL PSM, if the last instruction in a Block Statement is a `SELECT` statement the projection is now built on the server-side and returned. Block statements and SQL Methods are now behaving the same way.

```
BEGIN
  SELECT e.FirstName, e.LastName FROM Employee e
END;
```

- In SQL PSM, SQL methods can now be executed in the `SELECT` list

```
BEGIN
  SELECT e.FirstName, e.LastName, e.GetBonus(period) AS Bonus FROM Employee e
END;
```

- In SQL PSM, unlike in selections, accessing the 0th element in a relationship (i.e. `obj.MyRel(0).MyAttribute`) does not return an `OUT_OF_RANGE` error.
- In SQL, the `CURDATE()` built-in has been added. This built-in is a synonym for the `CURRENT_DATE()` built-in which returns the current date.
- In SQL, the `NOW()` built-in has been added. This built-in is a synonym for the `CURRENT_TIMESTAMP()` built-in which returns the current timestamp.
- In SQL, the compilation of a `SELECT` statement using an aggregate function (i.e. `max`) on the result of a `sublist()` call in the `SELECT` clause fails with a `INVALTYPE` error.
- In SQL PSM, in some cases the `CURRENT_TIMESTAMP` built-in may fail to return the correct current timestamp.
- In SQL PSM, in some cases comparing a declared object variable to `NULL` may cause the query compilation to fail with a syntax error.
- In SQL PSM, the compilation of a SQL method may lead to an `UNRESOLVED_METHOD` error. The message for this error code is not specific enough to rapidly pinpoint the root cause of the problem.
- Loading an ODL file with `mt_sdl` may fail with the 'relationship "myrel", which is not defined, does not exist in database "mydb"' error when the last class definition in the file contains the definition of multiple SQL methods.
- On Windows XP SP3, the Matisse 8.2.0 ODBC driver may fail to run properly.

Resolved in Matisse 8.2.0

- New optional database parameters have been added to improve the overall management of a Matisse database. The `DATEXTENDSZ` parameter defines the minimum datafile extension size used when the datafile is full. The `DATFULLINIT` parameter controls the datafile initialization size. The `DATINITSZ` parameter defines the minimum database size to be initialized before the server is online. The `MAXSRVLOGFILES` parameter defines the maximum version number of recycled server log files of the Matisse Server saved. The `MAXBKPLOGFILES` parameter defines the maximum version number of recycled backup log files of the Matisse Server saved.
- The execution of a SQL statement including a `LIKE-ESCAPE` clause using a backslash `'\'` character fails with a Matisse Syntax error.

```
select memberid from membership where memberid like '\%%' escape '\';
Error at line 1: 85a890a [MATISSE-E-SYNTAX_ERROR, MATISSE SQL syntax error]
```

Matisse 8.2.2 Release Notes

- In the Matisse Java binding, calling the ‘lookup’ method generated for primary key index may fail with the error “More than one object found with index 0x...”
- The C++ binding includes support for Matisse Event Notification mechanism. The C++ binding examples include a new project which demonstrates Matisse Event Notification mechanism.
- Matisse Data Transformation Services (`mt_dts`) may fail to create a many-to-many relationship directly from data stored in a CSV file if there is a read-only relationship involved.
- The `mt_dts` utility may fail to create a many-to-many relationship when its inverse relationship has not been explicitly declared.
- The `mt_dts` utility may fail to create an ordered one-to-many relationship directly from data stored in a CSV file.
- On Windows, on rare occasion a graceful database shutdown may fail to unregister the database from the Matisse port monitor.
- If the database server process is killed while extending a datafile, when the server is restarted the datafile auto-extend does not always trigger the datafile extension when the database is full.
- On Windows, Matisse client applications require `MATISSE_PORTMON_ADDR` to be defined (i.e. `tcp-7421`) even when Matisse TCP port monitor uses the default port number (7421).
- The Matisse license key installed on a laptop is not recognized when the network cable is not connected or when the Ethernet connection is using a wireless adapter.
- The customer license key format has changed in release 8.2. Matisse 8.2 does not recognize license keys issued for prior releases. Upon installation of Matisse 8.2, a license key with limited features is automatically issued.

Resolved in Matisse 8.1.4

- Matisse Enterprise Manager now displays a detailed description of each schema object in the Advanced Tab.
- C#, VB.NET, Java, C++ and Eiffel source code corresponding to the database schema classes can be generated from Matisse Enterprise Manager when selecting the new “Generate Code” menu item.
- In Matisse Enterprise Manager SQL Analyzer, detailed timing for a query execution is now reported in the message window.
- The Java binding includes support for Matisse Event Notification mechanism. The Java binding examples include a new project which demonstrates Matisse Event Notification mechanism.
- The Matisse Enterprise Manager as well as the `mt_sdl` utility generate Java source code for calling both server-side static and instance SQL methods without using a SQL statement, which seamlessly extends object-oriented programming to the database server. The Java binding examples include a new project which demonstrates the execution of SQL method directly from the corresponding Java method.
- Pentaho (a Business Intelligence suite) fails to query a Matisse database via Matisse JDBC driver.

- On Widows x64, 64-bit Java applications manipulating `MT_TIMESTAMP_LIST` may fail.
- Adding a new class to an existing database by importing an ODL file may delete SQL methods defined on existing classes.
- In Matisse Enterprise Manager SQL Analyzer, Blob values (`MT_BYTES`) in a table result-set are not properly displayed.
- In Matisse Enterprise Manager SQL Analyzer, the right-click popup menu for viewing multimedia data in a table result set is not enabled.
- On Widows x64, a SQL method returning an empty UTF16 character string may fail.

Resolved in Matisse 8.1.3

- Matisse installations now include a new demo (Friends) which shows how to search large graphs of entities stored in a Matisse database. This demo presents a quick and simple implementation of a breadth-first search algorithm using SQL methods to solve the single-source shortest path problem.
- Matisse Data Transformation Services (`mt_dts`) has now two new options tags to simplify the description of many-to-many associations. The `Name` option defined on the `PrimaryKey` tag and the corresponding `AssociatedName` option defined on the `IntermediateKey` tag make the relationship description more compact and more easy to read.
- With `mt_dts` you can now create a many-to-many relationship directly from data stored in a CSV file. The use of a CSV file is described in XML with three new tags: `IntermediateFile`, `FileName` and `IntermediateColumn`. This new option removes the need to create and load the temporary intermediate table previously needed to create a many-to-many relationship.
- In some rare cases on SPARC/Solaris 64-bit, user applications manipulating `MT_STRING_LIST` values and linked with `libmatisse.so.8.1` may fail with an "invalid address alignment" error.
- On SPARC/Solaris 64-bit, the Matisse Enterprise Manager may fail when selecting the Monitor node for an online database.
- In Matisse Enterprise Manager, the execution of a SQL select statement with navigation may return a "java.lang.NullPointerException" error if the first element in the resulting selection is NULL.


```
SELECT REF(Person.Friends) FROM Person
> java.lang.NullPointerException
```
- Matisse Enterprise Manager does not display Read-only relationships in a class description. In the Object Browser, the query results do not show Read-only relationship values.
- The execution of a SQL statement comparing the OID attribute to a non-constant value may return an empty result while an object with this OID exists in the database.
- In SQL PSM, in some cases a SQL method or a block statement may fail to compile when a Read-only relationship is accessed to view its successors.
- In SQL PSM, using the `COUNT ()` built-in function on a relationship may fail to execute with the `MATISSE-E-SQLRUNTIMEERR` error.


```
CREATE METHOD GetCount ()
```

```
RETURNS INT
FOR "Person"
BEGIN
    DECLARE res INT DEFAULT 0;
    SET res = COUNT(SELF.Friends);
    RETURN res;
END;
```

- The Matisse Eiffel binding does not build on Windows with Visual Studio 2005 without some edits in the Makefile.
- Support for Windows Vista x64 and Windows Server 2008 x64.

Resolved in Matisse 8.1.2

- The Matisse .NET binding now includes a new VS project which demonstrates Matisse support for unicode UTF16 in .NET.
- The .NET binding also includes a new VS project which demonstrates Matisse Event Notification mechanism.
- The .NET Object Manager utility `mt_dnom` now generates source codes for calling both static and instance SQL methods in the database server without using a SQL statement.
- In a SQL statement, an ODL file and a CSV file, you can now specify unicode character string constants for UTF16 using a ASCII character or a escape sequence `\uXXXX` that represents a UTF16-LE (Little Endian) character.
In SQL by specifying the `N` keyword as follows:
`N 'a text with \u00E9\u00F1 unicode sequences'`
In an ODL file as follows:

```
interface Person : persistent {
    attribute String UTF16 Alias = UTF16("a\u00E9b\u00F1c");
};
```


In a CSV file as follows:
`"Alias"`
`"a\u00E9b\u00F1c"`
- The `mt_sql` utility now shows UTF16 character strings with their escape representation.
- In Matisse Enterprise Manager SQL Analyzer, the object query results and table result sets now support right-click popup menus for viewing or editing data.
- In Matisse Enterprise Manager, the execution of a SQL select statement that returns a large number of objects may fail with the `'MATISSE-E-INVALIDSTREAM, <MtStream fd5f6f0> is not a valid stream for the selected database'` error.
- On Windows, Matisse Enterprise Manager does not release the file lock when the import of a CSV file fails preventing the file to be open by another program (i.e. notepad).
- On Windows, the line and position of the cursor in the Enterprise Manager SQL Analyzer Editor displayed in the status bar may be incorrect.

- Matisse SQL `INSERT` statement has been extended to build auto-increment attributes that can rely on the Matisse OIDs which are unique and incremental.

```
INSERT INTO movie (GUID, title) VALUES(CAST(OID AS INT), 'La Vie en Rose');
```

- SQL `LIKE` clause now also supports UTF16 pattern.

```
SELECT Name From Person WHERE Name LIKE N'a\u00E9'
```

- The execution of a SQL `INSERT` statement that includes a character string constant of more than 1.5 Mbytes in size may fail.

- SQL `INSTR(string1, string2, n, m)` built-in function does not return the correct position when searching backward with `n` is set to `-1`.

- In some cases the execution of a SQL method using a `WHILE` or `REPEAT` statement fails if the condition of the `WHILE` or `REPEAT` statement includes an `AND` operator.

```
BEGIN
while_loop:
  WHILE ((a IS NOT NULL) AND ( b > 1)) DO
    ...
  END WHILE;
END;
```

- The SQL compiler did not support 'always true' conditions such as `(1 = 1)` in a SQL method or a block statement.

```
BEGIN
while_loop:
  WHILE (1 = 1) DO
    ...
  END WHILE;
END;
```

- The SQL compiler did not support the definition of an `IF` statement with an empty `ELSE` clause in a SQL method or a block statement.

```
BEGIN
...
IF (a > 7) THEN
  SET res = 'found';
ELSE
  -- empty
END IF;
...
END;
```

- Deleting a large number of objects in a SQL block statement may fail with the `'MATISSE-E-OBJECTNOTFOUND, 0x0 not found'` error.

- In a very specific case, the database server does not restart with the `'GOM-F-INV, Invalid header in : 801FB5'` error.

- On Windows, the compilation of the C++ examples in a Visual Studio 2005 project is failing with the following error message:

```
c:\program files\matisse\include\matisse\mtconfig.h(14) : fatal error C1083: Cannot
open include file: 'iostream.h': No such file or directory
```

- Importing data from a CSV file that does not contain the column names in the first row fails.
- The `mt_xml` utility did not properly import “\r\n” escape sequences that were generated in the export process.
- On Windows, the SMListener service may stop when the user logs off.
- The `mt_sdl` utility may fail with the ‘MATISSE-E-OBJECTDELETED, 0x1188 has been deleted’ error when reloading an ODL file which describes meta-schema extensions that are not properly balanced. The schema description of meta-schema extensions must define both side of the relationship when this relationship is an extension of the meta-schema.

```
interface KClass : MtClass : persistent
{
    attribute String Description = String(NULL);

    relationship List<KAttribute> PublicAttributes
        inverse KAttribute::PublicAttributeOf;
};

interface KAttribute : MtAttribute : persistent
{
    attribute String Alias = String(NULL);

    relationship KClass PublicAttributeOf [0, 1]
        inverse KClass::PublicAttributes;
};

interface Document : persistent
{
    mt_instance_of KClass;
    mt_property Description = "My Document Class";

    attribute String Title
        mt_instance_of KAttribute
        mt_property PublicAttributeOf = {interface(Document)}
        mt_property Alias = "Document Title";

    mt_property PublicAttributes = {attribute(Document::Title)};
};
```

Resolved in Matisse 8.1.1

- The Matisse ADO.NET provider enables to execute multiple SQL statements in a single command:

```
DbCommand dbcmd = dbcon.CreateCommand();
dbcmd.CommandText =
    "SELECT REF(p) FROM Employee p WHERE p.ReportsTo IS NULL INTO emp_sel;\n" +
    "SELECT REF(p) FROM Manager p WHERE COUNT(p.Team) > 1 INTO mgr_sel;\n" +
    "SELECT REF(p) FROM SELECTION(emp_sel UNION mgr_sel) p;";
MtDataReader reader = (MtDataReader)dbcmd.ExecuteReader();
```
- Two new functions have been added to the SQL debugging module (`PSM_OUTPUT`). The `ENABLE` function enables calls to `PRINT`, `PRINT_LINE` and `PRINT_VARIABLE`. Calls to these functions are ignored if the `PSM_OUTPUT` module is not enabled. The `DISABLE` function disables calls to `PRINT`, `PRINT_LINE` and `PRINT_VARIABLE`, and purges the message buffer of any remaining information.

- In SQL PSM, the `SIGNAL` syntax now enables to assign `MESSAGE_TEXT` with an expression or a string variable

```
IF ... THEN
  -- raise an exception
  SIGNAL out_of_balance
  SET MESSAGE_TEXT = CONCAT('check balance for ', account);
END IF;
```

- In SQL PSM, the assignment of a newly created selection built with multiple selections from different classes could fail with the ‘`MATISSE-E-INCOMPTYPE`, Type of assignment source is incompatible with assignment target’ error.

- In SQL PSM, the `RETURN` statement did not support to call a SQL method that returns an object selection.

```
BEGIN
  DECLARE obj Movie;
  SELECT REF(o) INTO obj FROM Movie o WHERE o.OID = '0x1098';
  RETURN obj.GetMovieCast();
END;
```

- Changing a SQL method from an `INSTANCE` to a `STATIC` method or vice-versa had no effect.
- When compiling lengthy complex SQL methods, some compilation errors could return an incorrect error line.
- The C/C++ `MtSQLGetStmtInfo` function supports the new `MTSQL_STMT_ERRSTACK` option to provide users with a full 16 levels SQL method execution error stack trace. A similar option has been added to each object language binding. The `mt_sql` and `mt_emgr` utilities can now display a full SQL method execution error stack trace.
- In some cases, the ‘`MATISSE-E-SYSTEMERROR`, System error 0x9f4815’ error is returned when establishing a many-to-many relationship with `mt_dts` on classes where the intermediate table contains multiple duplicates for a given relationship.
- In Matisse Enterprise Manager SQL Analyzer, in some cases the result of a SQL method execution was not properly displayed. Object selection results could not be displayed in an object tree format.
- In Matisse Enterprise Manager SQL Analyzer, The ‘`MATISSE-E-NOTRANORVERSION`’ error could be returned when “Parse Query” is executed on some SQL DDL statements.
- In Matisse Enterprise Manager SQL Analyzer and Object Browser, The “Cancel execution” operation did not immediately cancel the execution of SQL methods containing an infinite loop.
- In Matisse Enterprise Manager SQL Analyzer, the SQL Editor did not always show the line and column position for the caret (cursor).
- In Matisse Enterprise Manager Schema, the SQL method tab supports SQL syntax highlighting. An option to show or hide the source code line number has been added as well.
- In Matisse Enterprise Manager Object Browser, the objects listed in the result panel now support the execution of SQL methods with a right-click on the object node.

- On Windows, in some cases Matisse Enterprise Manager could return an “<Host> is unreachable” error while trying to connect to a remote host.
- On Windows, loading ODL files using `MATISSE_CPP` with the Matisse Enterprise Manager could fail without providing a detailed enough error message.
- In some cases, loading ODL files using `MATISSE_CPP` could return inaccurate error line numbers.
- The `mt_sdl` utility could fail to load a SQL DDL file containing recursive SQL methods.
- The `mt_sdl` utility could fail to load a SQL DDL file with the error ‘`MATISSE-E-INVALID_PARAMETER` `MATISSE` SQL method "CONTAINS" has an invalid parameter (type mismatch for parameter 1)’ if the file contained a SQL method using the system method `CONTAINS` on a class that was not created in a previous transaction.
- The `mt_sdl` utility could fail to load ODL files if the file path contained spaces (' ') and `MATISSE_CPP` was set.

Resolved in Matisse 8.1.0

- The `mt_xml` utility returns always 1 on success while 0 is expected. The `mt_xml` utility returns 0 when the whole XML document has been stored into the database as new objects. However, `mt_xml` utility returns 1 if some elements in the XML document were not imported in the database, since they already existed in the database.
- In some cases, when using `mt_xml` to export a database, the following import reverses the order of some many-to-many ordered relationships.
- The ‘`mt_backup -d example write -f backup1.bkp -s 5G`’ command does not adjust the file size to the minimum of the remaining size to be backed up and the maximum size provided in the command line.
- The ‘`MATISSE-E-SYSTEMERROR, System error 0x8288102`’ can be returned when establishing relationships with `mt_dts` on classes where the types of the Primary key and Foreign key attributes are not identical.
- In some cases, the ‘`MATISSE-E-INVALSUCCSNB, Invalid number of successors 2 for relationship ...`’ error can be returned upon an `ALTER CLASS/ALTER RELATIONSHIP` statement.
- Version travel SQL queries do not always return the expected objects unless the version tag is created by a `COMMIT` statement.
- In Matisse Enterprise Manager, when restarting a database with a right-click on a database icon, the user interface seems to freeze until the database is online.
- A database with Access Control enabled cannot be restored from Matisse Enterprise Manager.
- On Linux, Matisse Enterprise Manager running for hours with auto-refresh enabled may generate an error which prevents from adding new entries to the Enterprise Manager log file.
- In some cases, the ‘`MATISSE-E-VERSIONMODE, Attempted to start a transaction in version mode`’ error is returned when executing the ‘Parse Query’ command from Matisse Enterprise Manager SQL Query Analyzer.

Resolved in Matisse 8.0.6

- Matisse Enterprise Manager auto-refresh option does not update the database schema tree to reflect the schema changes made by an external program or utility (i.e. `mt_sdl`).
- In Matisse Enterprise Manager, the Object Browser feature presents a Statistics panel to display the object count for each class in the database.
- “Add mirror” button and menu have been added to the Matisse Enterprise Manager Datafiles feature to ease the management of mirrored datafiles.
- The Matisse Enterprise Manager Datafiles feature now displays a Datafile Status column.
- On Windows, ‘`MATISSE-E-SQLSTACKOVF, MATISSE SQL stack over flow`’ could be returned when executing a SELECT statement.
- Establishing one-to-many or many-to-many relationships with `mt_dts` on classes with a few millions of objects could require to extend the data-files to a large size to complete in a reasonable amount of time.
- Matisse Data Transformation Services (`mt_dts`) did not allow import of attribute list types listed as one element per line.
- Matisse Data Transformation Services did not allow primary key-based updates. A new import option tag has been added: `AllowUpdates`.

Import Tag: `<allowUpdates>YES</allowUpdates>`

- Matisse Data Transformation Services (`mt_dts`) did not allow to import/export large binary data (`MT_BYTES`) and large text data (`MT_TEXT`) into files. By default, these two types are still imported/exported as field values. But using the `ColumnFromFile` and `ColumnToFile` option tags allow to manage the values into files.

Import Tag:

`<ColumnFromFile Directory="/import/documents">Document</ColumnFromFile>`

Export Tag:

`<ColumnToFile Directory="/export/documents" FilenameFormat="{DocIdent}.txt">Document</ColumnToFile>`

Resolved in Matisse 8.0.5

- Matisse Data Transformation Services (`mt_dts`) did not return a warning when a value is missing on the last row of the file and that the row does not include a newline character.
- On Windows, the environment system variables `MATISSE_CFG` and `MATISSE_LOG` defining alternate directories for the Matisse config and log files were ignored.
- Matisse Enterprise Manager could not display UTF16 String attribute values in the Object Browser.
- ‘`MATISSE-E-INTERNALERROR, RS PropVal opts`’ not cached error could be returned at commit when updating multiple relationships of an object after a class update including removing existing relationships and adding new ones.

- In some cases, an object could have a relationship pointing a non-existent object preventing any navigation or modification of the relationship of this object.

Resolved in Matisse 8.0.4

- On Windows the maximum size for a backup file slice was limited to 2Gbytes. This limit has been removed.
- On Windows, Matisse Enterprise Manager and Matisse Server Manager Listener did not always start properly when multiple versions of the Java Virtual Machine are installed on the machine and listed in the Path.
- Matisse Enterprise Manager could not export data from class 'Movie' in the 'Media' sample database
- In Matisse Enterprise Manager, when adding a mirror datafile with a file size not identical to the primary datafile, the invalid capacity error message was misleading.
- In Matisse Enterprise Manager Object Browser, the paging viewer did not always appear when a SQL query returned a large collection of objects.
- In Matisse Enterprise Manager Object Browser, an Advanced tab has been added to make both standard and advanced query builders more independent of each other.
- Matisse Data Transformation Services (`mt_dts`) did not support importing/exporting multimedia data into files. Two XML options tags have been added: `ColumnFromFile` and `ColumnToFile`

Import Tag:

```
<ColumnFromFile Directory="/import/images">Photo</ColumnFromFile>
```

Export Tag

```
<ColumnToFile Directory="/export/images" FilenameFormat="img_{LastName}.jpg">Photo</ColumnToFile>
```

- Matisse Data Transformation Services (`mt_dts`) did not allow to link classes with primary/foreign keys composed of multiple columns.
- SQL DDL script generated with `mt_sdl` did not add double-quote characters (") around every attribute and class names which could collide with SQL reserved keywords (i.e. Group should be "Group")
- in some cases, a SQL method returning a large object collection could include a null object.
- 'MATISSE-E-OBJECTNOTFOUND, 0x0 not found' error could be returned at commit when updating an object with multiple ordered relationships with a very large number of successors.

Resolved in Matisse 8.0.3

- In the Enterprise Manager tool, icons were not displayed in the Index Name and Criteria[1-4] columns of the Indexes table as well as in the Entry Point Name and Attribute Name columns of the Entry Point Dictionaries table when presenting the properties of a Class.
- In some cases, accessing a `MT_STRING_LIST` value using Matisse C++ Binding on a 64-bit platform could lead to a memory corruption.

- All tools and utility commands print out a message which distinguishes clearly the 64-bit edition from the 32-bit edition.

```
$ mt_emgr -V  
MATISSE Enterprise Manager x32 Version 8.0.3 (32-bit Edition) - Sep 21 2006.  
(c) Copyright 1992-2006 Matisse Software Inc. All rights reserved.
```

```
$ mt_emgr -V  
MATISSE Enterprise Manager x64 Version 8.0.3 (64-bit Edition) - Sep 21 2006.  
(c) Copyright 1992-2006 Matisse Software Inc. All rights reserved.
```

- Added `--full` option to the `mt_server checklicense` command to provide a complete description of the installed license.

Resolved in Matisse 8.0.2

- `'MATISSE-E-SYSTEMERROR, System error 0x846817a'` could be returned when opening a connection to a remote server protected by a firewall.
- Some nodelocked license key installed on Windows machines with multiple active Ethernet cards can prevent database servers to restart after a reboot of the host machine.

7 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.