Matisse® 8.4.8
Release Notes

August 2011

Matisse 8.4.8 Release Notes

PDF generated 3 August 2011

Contents

1 New Features in Matisse 8.4

1.1 Overview

The Matisse 8.4 release introduces new features and major enhancements in the Matisse product line:

- *Matisse In-Memory Database*, the in-memory version of Matisse DBMS, has been added to the product line.
- *Matisse Database Modeler* is a modeling tool to help you manage your database schema. The Matisse Modeler allows you to draw and maintain a UML-like diagram of your application model.
- *Matisse Lite*, the embedded version of Matisse DBMS, can be installed as a product on Windows to ease the development of embedded applications.
- The *Matisse Enterprise Manager* has been enhanced to improve the developers experience. The new *Task Scheduler* allows administrators to manage recurring administration tasks on local and remote database servers. The Import Database Schema and Export XML features have been extended to provide more options to the developers and administrators.
- Some of the Matisse command line utilities have received new options to ease database administration.
- The Matisse Eiffel binding has been redesigned to take advantage of the features of Eiffel 6.6 and to improve the familiarity with the APIs of other major bindings.
- The Matisse Python binding has been redesigned to take advantage of Python new features and to improve the familiarity with the APIs of other major bindings.

1.2 Matisse In-Memory Database

Matisse In-Memory Database is the in-memory version of Matisse DBMS. Matisse In-Memory is a 64-bit only version of Matisse managing all data in-memory.

Matisse In-Memory Database is a memory-optimized database that empowers applications with the instant responsiveness and very high throughput required by real-time industries. Matisse In-Memory Database operates on databases that fit entirely in physical memory using standard Matisse APIs. Matisse In-Memory Database exploits this property for breakthrough performance.

Matisse In-Memory Database was designed from the ground up to leverage large memory size on 64-bit commodity hardware, along with the processing scalability provided by multi-core, multi-processor architectures. By managing all data in memory and using algorithms tailored for that environment, Matisse In-Memory Database can operate much more efficiently, and thus offer dramatic improvements in performance. Matisse In-Memory Database uses disks for persistence and recovery rather than as the primary database storage location. Matisse In-Memory Database conforms to the atomicity, consistency, isolation, and durability (ACID) properties of data management systems. Durability is achieved by periodically updating a disk image of the database, called a checkpoint. The database is checkpointed to disk automatically based on user-defined frequency and transaction volume.

The decreasing cost of memory-rich 64-bit computing platform is driving the in-memory database technology's use simulation, analytics, in trend and pattern discovery, caching and other systems that demand instantaneous sorting, retrieval and manipulation of massive databases.

Matisse In-Memory 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 true real-time application by simply restarting your database with a Matisse In-Memory database server.

1.3 Matisse Database Modeler

Matisse Database Modeler is a modeling tool to help you manage your database schema. The Matisse Modeler allows you to draw and maintain a UML-like diagram of your application model, then save it into a Matisse database. The Database Modeler main features includes the generation of stub classes of the main supported programming languages, the ability to print diagrams and save them as images.

1.4 Matisse Lite

Matisse Lite, the embedded version of Matisse DBMS, is a compact library that implements the server-less version of Matisse.

In this release, all the Matisse libraries included in the delivery are now specific to the Matisse Lite product.

The Matisse Enterprise Manager tool has been improved to better manage a group of Lite databases.

On Windows, the installer has been updated to install Matisse Lite as a product to ease the development of applications.

1.5 Enterprise Manager Tool

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

Tasks Scheduler

The `Tasks` node has been added to the `Management` group in the database tree hierarchy. The Task scheduler allows administrators to manage recurring administration tasks on local and remote database servers.

Import Schema

Two new dialog windows have been added to import database schema respectively in ODL or in SQL-DDL. Each dialog includes a text viewer to preview the database schema prior to loading it into the database.

Export XML

The new Export Xml Dialog window gives access in the Enterprise Manager to all the export options provided by the `mt_xml` command line utility.

SQL Query Analyzer

The SQL Query Analyzer shows clear distinct values for `NULL`. The `NULL` type is displayed as **`NULL`** (uppercase, bold font), the null pointer values is displayed as **`null`** (lowercase, bold font) and all other `NULL`/`null` values are character strings.

1.6 Database Utility Commands

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

mt_server checkpoint

The `mt_server` utility with the `checkpoint` command allows you to create a checkpoint file on disk for a running in-memory database.

```
mt_server [OPTIONS] checkpoint [-h]
-h, --help       Display this help and exit
```

mt_server list

The `mt_server list` command with the `-s` option provides the list of offline databases.

```
mt_server list -h

Usage: mt_server [OPTIONS] list [-h]
  -r, --running  List all running databases (default)
  -s, --stopped  List all stopped databases
  -h, --help     Display this help and exit

mt_server list -s
2server(s) offline
           Security InMemory  CacheSize  Capacity
analysisim      Off      Yes        64M       1G
```

```
example          Off      No          64M        5M
media            Off      No          64M        4M
northwind        Off      No          64M       25M
reports          Off      No          64M       30M
```

mt_server extendcache

The `mt_server` utility with the `extendcache` command provides the `-o` option which allows you to extend the object table cache size.

```
mt_server [OPTIONS] extendcache [-p|-o] -s <size>[GMK] [-h]
  -p, --page      Page cache size (default)
  -o, --object    Object table cache size
  -s, --size=...  New cache size (M is default)
  -h, --help      Display this help and exitmt_server [OPTIONS]
```

mt_server setrunfrequency

The `mt_server` utility with the `setrunfrequency` command allows you to change on an online database the run frequency of the various automatic operations.

```
mt_server [OPTIONS] setrunfrequency [-a|-c] -f <freq>
  -a, --autovc    Set auto-collect run frequency (default)
  -c, --clearot   Set clear object table run frequency
  -f, --freq=...  run frequency in seconds
  -h, --help      Display this help and exit
```

1.7 Matisse XML Manager

Matisse XML Manager has been enhanced to improve the export performance of very large XML files.

mt_xml export

The `mt_xml` utility with the `export` command allows you to export the database in multiple files in parallel while controlling the prefetching of objects from the database server.

```
C:\> mt_xml
MATISSE XML Manager x32 Version 8.4.1.0 (32-bit Edition) - Oct 28
    2010.
(c) Copyright 1992-2010 Matisse Software Inc. All rights reserved.

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

[...]

   -parallel <n>     : Exports data with <n> tasks running in parallel.
```

```
                              The XML data is exported into multiple XML files
                              named <filename>_xds_<documentid>.xml.

-prefetch <n>          : Specifies the number of objects to be prefetched
                          when exporting data. The default value is 128.
                          The values range between 1 and 128.
```

1.8 Eiffel Binding

Eiffel 6.6 Support

Matisse Eiffel binding has been redesigned for Eiffel 6.6 and relies on Eiffel 6.6 features to provide the implementation of a new API familiar with the APIs of the Java, C++ and .NET bindings.

Examples

Matisse Eiffel binding examples have been redesigned to further reduce the programming learning curve and to enhance the developers experience.

1.9 Python Binding

Python 3 Support

Matisse Python binding has been redesigned for Python 3 and relies on Python new features to provide the implementation of a new API familiar with the APIs of the Java, C++ and .NET bindings.

Examples

Matisse Python binding examples have been redesigned to further reduce the programming learning curve and to enhance the developers experience.

1.10 Database Configuration Parameters

New optional database parameters have been added to support in-memory databases, to improve the control the frequency of the automatic version collection operation as well as to better control the size of the of the multi-version object table memory cache.

DATINMEMORY

This parameter defines the primary location of the datafiles. Specifying a value of 1 enables an in-memory database. The datafiles are in-memory and disks are used for checkpointing the database. When it is set to 0, the datafiles are located on disks.

AUTOCOLLECTFREQ

This parameter defines the run frequency of the automatic version collection operation. This parameter is expressed in seconds. The minimum value is 5 seconds, the maximum size is 360 seconds.

OBJTABLESIZ

This parameter defines the maximum size of the multi-version object table memory cache. The pages in the object table cache are only allocated when required. This parameter is expressed megabytes (M suffix), gigabytes (G suffix). Specifying a value of 0 disables the control of the maximum size.

OBJTABCLRFREQ

This parameter defines the run frequency of the object table clearing operation. This parameter is expressed in seconds. The minimum value is 3 seconds, the maximum size is 120 seconds.

2 Compatibility with Previous Releases

2.1 Matisse 8.4 Data Migration

Matisse Server 8.4 comes with several changes in the data format. You must use the `mt_xml` tool for converting an existing database (8.3.x or prior) into the 8.4 format.

Step 1

Before installing 8.4.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 8.4.x on your machine and then restore the schema and the data as follows:

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

2.2 Client Connections

Only 8.4.x clients may be used with 8.4.x servers.

The clients for earlier releases of Matisse are incompatible with the 8.4.x server. Consequently, you must upgrade any older clients to 8.4.x before attempting to access an 8.4.x server.

3 Platform-Specific Topics

3.1 Linux

The following Linux distributions on x86 (32-bit) and x86_64 (64-bit) chips families are supported:

- CentOS
- Debian
- Fedora Core
- Red Hat Enterprise Linux
- SUSE Linux Enterprise Server
- Ubuntu

Any other Linux distributions, where Matisse 8 has not been tested, require Linux kernel 2.6.9 on systems based on x86 (32-bit) or x86_64 (64-bit) chips families.

3.2 MacOS

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

3.3 Solaris

Support for Solaris 10 on x86 (32-bit) and x86_64 (64-bit) chips families.

The Solaris 10 on SPARC with 32-bit kernel and 64-bit kernel is available upon request.

3.4 Windows

Support for Windows (XP/2003/2008/Vista/7) on systems based on x86 (32-bit) and x86_64 (64-bit) chips families.

4 Update History

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

Resolved in Matisse 8.4.8

- In the Enterprise Manager Sql Analyzer, the execution of a `SELECT` statement with a `LIMIT / OFFSET` clause returning no objects while some objects qualify raises a Java exception.
- In the Enterprise Manager Sql Analyzer, The qualified object count is not reported in the message window when it differs from the selected object count.
- On Windows, the `mt_server` utility with the `recyclelog` command does not correctly create the historical version of the recycled log file.
- On Windows, the `mt_server` utility with the `list` command may report the generic error message `'STS-FAILED Unknown message status=8358094'` instead of a specific error message.
- When the database is full and the database auto-extend option is turned off, the shutdown command does not return the appropriate error to indicate that graceful shutdowns cannot be executed unless the database size is increased.
- In some cases, the compilation of a SQL block statement or a SQL method updating objects from a class with a large number indexes are associated with may fail with the error message:
  `864802a: MATISSE-E-ARRAYTOOSMALL, Array too small - n elements needed`
- In rare cases, the execution of a SQL query with multiple `LIKE` clauses may fail returning the error message:
  `MATISSE-E-SYSTEMERROR, System error 0x846814a`
- On Windows 64-bit in some cases, the compilation of a SQL `SELECT` statement with over 40 `OR` clauses may fail.
- In the C++ binding, exceptions may be raised with an error message that does not match the error code.
- In some cases, after the partial restore of a multi-file backup, the following restart of the database server does not properly report the error indicating that the database is inconsistent.
- On Windows, a Matisse In-Memory server with multiple in-memory datafiles may fail with a segv error when destroying the inconsistent checkpoint files on disk.

Resolved in Matisse 8.4.7

- New optional database configuration parameters have been added to improve the control the frequency of the automatic version collection operation as well as to better control the size of the of the multi-version object table memory cache.
- The `mt_server` utility with the `extendcache` command provides the `-o` option which allows you to extend the object table cache size.

- The `mt_server` utility with the `setrunfrequency` command allows you to change on an online database the run frequency of the various automatic operations.
- On Windows, using usernames with white space character (`' '`) may prevent the Enterprise Manager as well as the `mt_server` command to start databases.
- The generated source code for the `Python` binding has been slightly improved.
- The generated source code for the `PHP` binding has been slightly improved.
- Compiling the C++ binding files with the latest MS-Windows compilers as well as with the latest GNU compilers reports unnecessary warnings.
- The `mt_sql` command line may not report the correct error message when trying to connect to an offline database.
- The SQL query optimizer does not always select the optimal index for `LIKE` clauses when multiple multi-segments indexes exist.
- The execution of a SQL query with a `LIKE '%'` clause may fail to return the expected result when the selection contains a very large number of objects.
- In some cases, the execution of a SQL query with a unicode `LIKE` clause may fail to return the expected result.
- In rare cases, the SQL built-in `ROUND(X[,D])` used with only one parameter may fail to truncate the result to 0 digit.

Resolved in Matisse 8.4.6

- The Matisse Python binding has been redesigned to take advantage of Python new features and to provide an API very similar to the API of the C++, Java and .NET bindings.
- In the Enterprise Manager, the index count reported in the Schema Index tab is not always correct.
- In some cases, the XML import may fail to report an error when creating an index key for an attribute with a character string value that does not fulfill the maximum size constraints.
- In the Sql Editor of the Enterprise Manager, the execution of a large number of SQL statements modifying the database schema may generate unexpected errors such as:
  `MATISSE-E-INVALDATAACCESSMODE, Invalid data access mode.`

Resolved in Matisse 8.4.5

- The C API error functions have been extended to receive a `MtContext` argument:

```
MtString MtCtxError(MtContext ctx);
void  MtCtxPError(MtContext ctx, MT_CONST MtChar* comment);
MtOid MtCtxGetInvalidObject(MtContext ctx);
MtSTS MtCtxMakeUserError(MtContext ctx, void* error, MT_CONST MtChar* errorString);
void* MtCtxGetUserError(MtContext ctx);
MtBoolean MtCtxCheckErrorP(MtContext ctx, MtSTS status);
MtSTS MtCtxGetCheckErrorStatus(MtContext ctx);
```

- The following two functions have been removed from the C API:
  ```
  MtSTS MtCtxGetContextOffset(MtContext ctx, MtSize *offset);
  MtSTS MtCtxGetContextFromOffset(MtContext *ctx, MtSize offset);
  ```
- The execution of a `SQL` query may return an incorrect result set when the `WHERE` clause contains multiple `AND` statements and that not all the properties used for comparison are indexed.
- The execution of a `SQL` query may fail to return a result in a reasonable amount of time when the `GROUP BY` clause is based upon a 2 level deep Composition relationship.
  ```
  SELECT patient.clinic.location, count(*)
  FROM treatment
  GROUP BY patient.clinic.location;
  ```
- The execution of a `SQL` query with a `GROUP BY` clause may return an unexpected result when some of the attribute values are equal to the attribute default value.
- The 64-bit version of the Matisse `Java` binding has been improved to provide in some cases better scalability for applications a large pool of connections.
- On Windows 64-bit, in some rare cases the 64-bit Matisse `.NET` binding may corrupt the memory of the unmanaged code layer of the binding which could lead to unpredictable behaviors.
- The Matisse `C++` binding no longer supports the `MtHiddenContext` which relies upon the Matisse C functions with no explicit connection context. Only the implementation with an explicit connection context remains supported.
- The Matisse `PHP` binding has been improved to provide in some cases better scalability for applications a large pool of connections.
- The Matisse `Eiffel` binding has been improved to provide in some cases better scalability for applications a large pool of connections.
- In some cases, the `mt_xml` utility with the `-parallel` option may fail to properly generate the XML document files.

Resolved in Matisse 8.4.4

- The `MtCtxSkipObjects` function has been added Matisse C APIs to simplify skipping objects in an object enumeration stream:
  ```
  MtSTS MtCtxSkipObjects(MtContext ctx, MtStream stream, MtSize *numObjects)
  ```
- All the language bindings implement the `skip()` method on an object enumerator with the new `MtCtxSkipObjects()` C function.
- In the Java binding the execution of `MtObjectIterator.skip(int n)` with `n` set to `Integer.MAX_VALUE` may fail with `MATISSE-E-NULLPOINTER, Null pointer`.
- In the .NET binding the execution of `MtObjectEnumerator.Skip(Int32 n)` with `n` set to `Int32.MaxValue` may fail with `MATISSE-E-NULLPOINTER, Null pointer`.

- In a very specific case on a multi-CPU machine, when the automatic version collection triggers an automatic extension of the datafile, the server process CPU usage may increase slowing down significantly client requests currently in process.
- In the Enterprise Manager, the fonts Editor, Result, Message and Preview windows changed during a session are not properly restored when a new session is started.
- The compilation of a `SQL` method returning a class instance may fail when the class `OID` is greater than `0x7FFFFFFF`.
- When the `OID` upper limit is reached, a unique error is now returned (`MATISSE_OIDEXHAUSTED`). The server log file now displays a more specific warning message when the OID surrogate reaches a predefined threshold. A XML migration of the database restores the OID surrogates to the lower end.

Resolved in Matisse 8.4.3

- The Matisse Database Modeler has been added to the Matisse database product line.
- In the Enterprise Manager, the Preferences menu-item in the View menu is now enabled. The Preferences dialog allows you to change the Fonts of the Editor, Result, Message and Preview windows.
- In the Enterprise Manager, when the `AUTOCOLLECT` configuration option is disabled, the Database properties dialog does not always reflect the change while effective in the configuration file.
- The SQL query optimizer does not always select the optimal index when multiple multi-segments indexes exist in a hierarchy of classes.
- In a very specific case, a full backup may not restore the database with the appropriate layout making some objects no longer accessible.

Resolved in Matisse 8.4.2

- In the C API 2 new functions are added to complement the `MtIsPredefinedObject` and `MtCtxIsPredefinedObject` functions:
```
MtIsMetaSchemaObject(MtBoolean* meta, MtOid object)
MtCtxIsMetaSchemaObject(MtContext ctx, MtBoolean* meta, MtOid object)
```
- In the .NET binding, 2 new methods are added to the `MtDatabase` to retrieve the object that was involved in an error during the commit phase (i.e. `MATISSE_DEADLOCK, MATISSE_DEADLOCKABORT`):
```
public int MtDatabase.GetInvalidMtOid()
public MtObject MtDatabase.GetInvalidMtObject()
```
- In the Java binding, 2 new methods are added to the `MtDatabase` to retrieve the object that was involved in an error during the commit phase (i.e. `MATISSE_DEADLOCK, MATISSE_DEADLOCKABORT`):
```
public final int MtDatabase.getInvalidMtOid()
public final MtObject MtDatabase.getInvalidMtObject()
```
- In the Enterprise Manager, in some cases user-defined classes can be listed under the Meta-Schema node instead of under the Schema node.

- In the Java binding, the constant values documentation page (`java/api/constant-values.html`) is missing.
- In the Java binding, in some cases the execution of a generated method matching a SQL method defined on the persistent class may fail with `MATISSE-E-INVALSTMT, invalid statement`
- In some cases with heavy multi-user usage involving numerous connection and disconnection activities, the client programs may receive the `NET-E-SRVCONFAILED, Connection to server failed` error.

Resolved in Matisse 8.4.1

- The `mt_server list` command with the `-s` option provides the list of offline databases.
- The `mt_xml export` command receives 2 new options `-prefectch` and `-parallel` to speed up the export of large databases.
- In the Enterprise Manager, the menubar of the Monitor tab includes a new button to allow the execution of a version collect operation.
- In the Enterprise Manager, the new popup menu (`"View Data"`) has been added to the Classes table of the Schema viewer tab.
- In the Enterprise Manager Sql Analyzer, the header of the result table does not redraw properly when the table has a large number of columns and that it is horizontally scrolled back and forth.
- In the Enterprise Manager Object Browser, clicking on the Refresh button on the database statistics tab may freeze the UI when the statistics on a very large database are recomputed.
- When an invalid value is set for connection options `MT_MEMORYTRANS_BUFSZ` or `MT_NETWORKTRANS_BUFSZ`, the error message returned is meaningless.
- In some specific cases a server-side SQL update may fail to update the attribute values when this attribute is part of an index.

Resolved in Matisse 8.4.0

- The Matisse In-Memory Database server has been added to the Matisse database product line.
- Matisse Lite defines a new set of libraries clearly distinct from the ones of the standard Matisse DBMS.
- Matisse Lite for Windows is installed as a product through an installer program.
- The Enterprise Manager includes a task scheduler to ease the management of recurring administrative jobs.
- The SQL Query Analyzer in the Enterprise Manager now show clear distinct values for `NULL`. The `NULL` type is displayed as **`NULL`** (bold font), the null pointer values is displayed as **`null`** (bold font) and all other `NULL`/`null` values are character strings.

- The Enterprise Manager provides new dialog windows to import schema in ODL or in SQL-DDL that includes a text viewer to preview the database schema prior to loading it into the database.
- The Enterprise Manager now includes a new Xml Export Dialog window that provides more options for exporting data in Xml format.
- The Enterprise Manager can display the historical log files of all main components that includes the Enterprise Manager itself, the port monitors, the server manager listener, as well as offline databases through the addition of a Log tab at the component node level.
- The Enterprise Manager now preserves the location of the split-panes defined by the user between sessions.
- The Import Data Dialog in the Enterprise Manager now tries to best guess the field delimiter used in the CSV file to be loaded.
- The Enterprise Manager now try to provide a default filename when saving a CSV, XML, ODL, or SQL DDL file.
- The `mt_server` command with the `checkpoint` option creates a checkpoint file on disk for a running in-memory database.
- The Matisse Eiffel binding has been redesigned to take advantage of the new features of Eiffel 6.6 and to provide an API very similar to the API of the C++, Java and .NET bindings.
- In some cases, the splash screen of the Enterprise Manager may take up to 2 seconds to appear leading the user to believe the program did not start.
- In some cases, the Enterprise Manager seems slow to show the database schema objects (i.e. classes, indexes, etc.) while it is fast to retrieve the Meta-Schema objects.
- In some cases, the Enterprise Manager seems unresponsive when the tree-node objects are refreshing.
- On Windows Vista, a server with security on started by the Enterprise Manager limits the server accessibility to only programs with System privileges.
- In a specific case, the Enterprise Manager crashes when adding a `MT_STRING` attribute with a default value to a class on a database loaded with millions of objects.
- in some cases, loading an ODL file may fail to create the database schema reporting the "`MATISSE-E-INVALKEYLEN, Invalid key length`" error when the schema includes `UTF16 String` attributes and indexes defined in a specific order.
- Loading an Xml file into the database may fail if the file contains objects with attributes of type `MT_STRING_LIST` and that the list elements are `NULL` pointer values.
- The Matisse SQL APIs do not distinguish between the `NULL` type and a value with a `null` pointer value for data types that support null pointer values (i.e `LISTs`, `MT_STRING`, `MT_TEXT`, `MT_STRING_LIST` where element can be `null` pointers). The C `SQLGetRowTypeValue` functions have been updated to provide the necessary information:

```
MtSTS MtCtxSQLGetRowTypeValue(MtContext ctx, MtStream stream, MtSize colnum,
MtType* type, MtType *rowType, MtCharacterEncoding* encoding, void* value, MtSize*
sz, MtBoolean* loadedVal, MtBoolean* nullptrVal)
```

```
MtSTS MtCtxSQLMGetRowTypeValue(MtContext ctx, MtStream stream, MtSize colnum,
MtType* type, MtType *rowType, MtCharacterEncoding* encoding, void** value, MtSize*
sz, MtBoolean* loadedVal)
```

- A Java application using a Matisse Lite database may fail to load the Matisse Java Lite library when querying the database through Matisse native JDBC.

5 Documentation

Matisse documents available on the Web

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

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

Documents included with Matisse standard installation

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

Open source bindings

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

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