Matisse® 9.1.12
Release Notes

December 2019

Matisse 9.1.12 Release Notes

PDF generated 30 December 2019

Contents

1 New Features in Matisse 9.1

1.1 Overview

The Matisse 9.1 release introduces new features and major enhancements in the Matisse product line:

- Matisse DBMS provides full distributed XML-based asynchronous replication to complement the existing synchronous transactional replication capability.
- Matisse new database activity monitoring features help database administrators monitor the performance and availability of databases on a distributed network.
- Direct-io datafiles provide the performance and reliability that we normally see only in raw devices, but also hides the complexity of managing raw devices.
- A set of new optional database configuration parameters have been added to fine tune the performance of production databases.
- Services, tools and utilities transparently support IPv6.

1.2 Asynchronous Replication

Matisse XML-based Snapshot Replication is a full distributed asynchronous replication. The Publisher-Subscribers model applies to describe how incremental changes are propagated from the Publisher (master database) to Subscribers (replica databases) as they occur. Snapshot replication typically starts with a full data snapshot of the Publisher database. As soon as the initial snapshot is taken, subsequent data changes made at the Publisher are delivered on demand to the Subscribers. All data snapshots (full and increment) are published into XML documents.

The Subscriber initiates the replication by requesting a full data snapshot of the Publisher database. The XML documents produced are equivalent to a full XML export of the database. Subsequent requests from the Subscriber produce data snapshot increment reflecting the net data change since the previous request.

The Subscriber database is synchronized with the Publisher when all the data snapshots are loaded. The data snapshots must be loaded in the order they have been produced.

The production of data snapshots into XML document files give database administrators a great latitude to design the most appropriate replication workflow. The XML format is simple, extensible and universal and XML documents compress very well which is ideal for network transfers. The Enterprise Manager Task Scheduler is well suited to automate the replication workflow.

Production environments that require a minimum downtime can benefit from Snapshot replication to streamline major software and hardware upgrades.

Example

The replica database (Subscriber) replicates the master database (Publisher) content using Snapshot replication.

On the Publisher side, the first step requires to export the database schema of the publisher database and to establish the replication with a full data snapshot.

```
$ mt_sdl -d master@localhost export --odl -f masterDbSchema.odl

$ mt_xsr -d master@localhost --verbose=2 publish -f masterDb_01f.xml
    -n xsrExample --full
  [INFO] task #1 writing masterDb_01f_xsr_ia001.xml
  [INFO] task #1 writing masterDb_01f_xsr_ir002.xml
  [STAT] Number of top-level objects published: 8
  [STAT] Number of object insert published: 8
  [STAT] Number of object update published: 0
  [STAT] Number of object delete published: 0
  [OPTN] Number of prefetch objects: 128
  [OPTN] XML data with OID xml attribute: YES
  [OPTN] Media data into external files: NO
  [OPTN] Namespace: xsrExample
  [OPTN] XML data file I/O mode: stream
  [TIME] Start schema info building:       15:48:29.703    Elapsed
    00:00:00.000
  [TIME] End schema info building  :       15:48:29.704    Elapsed
    00:00:00.000
  [TIME] Start extracting:                 15:48:29.686    Elapsed
    00:00:00.000
  [TIME] End extracting  :                 15:48:29.708    Elapsed
    00:00:00.022

$ mt_xsr -d master@localhost describe --publisher
XML-based Snapshot Replication publisher on database master at time 6
  Publisher #1
    Publisher name:      master@localhost
    Snapshot type:       full (#1)
    Version name:        MTXSR00001086_00000001_00000005
    Version time:        6
    Publisher namespace:  xsrExample

$ mt_xsr -d master@localhost describe -f masterDb_01f.xml
XML-based Snapshot Replication Document:
  Filename:         masterDb_01f.xml
  Publisher:        master@localhost
  Generation date:   2013-05-02 15:48:29
  Snapshot type:    full (#1)
  Version name:     MTXSR00001086_00000001_00000005
  Version time:     6
  Namespace name:   xsrExample
  Insert count:     8
  Update count:     0
  Delete count:     0
```

On a regular basis data snapshot increments are produced.

```
$ mt_xsr -d master@localhost --verbose=2 publish -f masterDb_01i1.xml
    -n xsrExample --increment
  [INFO] task #1 writing masterDb_01i1_xsr_ia001.xml
  [INFO] task #1 writing masterDb_01i1_xsr_ir002.xml
  [INFO] task #1 writing masterDb_01i1_xsr_ua003.xml
  [INFO] task #1 writing masterDb_01i1_xsr_ur004.xml
  [STAT] Number of top-level objects published: 6
  [STAT] Number of object insert published: 2
  [STAT] Number of object update published: 4
  [STAT] Number of object delete published: 0
  [OPTN] Number of prefetch objects: 128
  [OPTN] XML data with OID xml attribute: YES
  [OPTN] Media data into external files: NO
  [OPTN] Namespace: xsrExample
  [OPTN] XML data file I/O mode: stream
  [TIME] Start schema info building:       15:55:10.271    Elapsed
    00:00:00.000
  [TIME] End schema info building  :       15:55:10.280    Elapsed
    00:00:00.009
  [TIME] Start extracting:                 15:55:10.258    Elapsed
    00:00:00.000
  [TIME] End extracting  :                 15:55:10.292    Elapsed
    00:00:00.033

$ mt_xsr -d master@localhost describe --publisher
XML-based Snapshot Replication publisher on database master at time 11
  Publisher #1
    Publisher name:       master@localhost
    Snapshot type:        increment (#2)
    Version name:         MTXSR00001086_00000002_00000009
    Version time:         10
    Publisher namespace:  xsrExample

$ mt_xsr -d master@localhost describe -f masterDb_01i1.xml
XML-based Snapshot Replication Document:
  Filename:          masterDb_01i1.xml
  Publisher:         master@localhost
  Generation date:   2013-05-02 15:55:10
  Snapshot type:     increment (#2)
  From version name: MTXSR00001086_00000001_00000005
  Version name:      MTXSR00001086_00000002_00000009
  Version time:      10
  Namespace name:    xsrExample
  Insert count:      2
  Update count:      4
  Delete count:      0
```

On the Subscriber side, the first step requires to import the database schema of the publisher database and to establish the replication by loading the full data snapshot.

```
$ mt_sdl -d replica@localhost import --odl -f masterDbSchema.odl

$ mt_xsr -d replica@localhost --verbose=2  subscribe -f
    masterDb_01f.xml -n xsrExample
  [INFO] task #1 loading masterDb_01f_xsr_ia001.xml
  [INFO] task #1 loading masterDb_01f_xsr_ir002.xml
```

```
  [STAT] Number of top-level xml objects read: 8
  [STAT] Number of objects created: 8
  [STAT] Size of oid mapping table: 0.01 MB
  [OPTN] Namespace origin: xsrExample
  [OPTN] Namespace destination: xsrExample
  [OPTN] Number of xml objects parsed at once: 256
  [OPTN] Number of objects per transaction: 20480
  [TIME] Start loading:                  16:06:05.519    Elapsed
    00:00:00.000
  [TIME] End loading  :                  16:06:05.526    Elapsed
    00:00:00.006

$ mt_xsr -d replica@localhost describe --subscriber
XML-based Snapshot Replication subscriber on database replica at time
    8
  Subscriber #1
    Publisher name:       master@localhost
    Snapshot type:        full (#1)
    Version name:         MTXSR00001086_00000001_00000005
    Version time:         6
    Publisher namespace:  xsrExample
    Subscriber namespace: xsrExample
```

When a new data snapshot increment is available, it can be loaded into the Subscriber database.

```
$ mt_xsr -d replica@localhost --verbose=2  subscribe -f
    masterDb_01i1.xml -n xsrExample

  [INFO] task #1 loading masterDb_01i1_xsr_ia001.xml
  [INFO] task #1 loading masterDb_01i1_xsr_ua003.xml
  [INFO] task #1 loading masterDb_01i1_xsr_ir002.xml
  [INFO] task #1 loading masterDb_01i1_xsr_ur004.xml
  [STAT] Number of top-level xml objects read: 8
  [STAT] Number of objects created: 4
  [STAT] Size of oid mapping table: 0.01 MB
  [OPTN] Namespace origin: xsrExample
  [OPTN] Namespace destination: xsrExample
  [OPTN] Number of xml objects parsed at once: 256
  [OPTN] Number of objects per transaction: 20480
  [TIME] Start loading:                  16:06:36.903    Elapsed
    00:00:00.000
  [TIME] End loading  :                  16:06:36.915    Elapsed
    00:00:00.012

$ mt_xsr -d replica@localhost describe --subscriber
XML-based Snapshot Replication subscriber on database replica at time
    12
  Subscriber #1
    Publisher name:       master@localhost
    Snapshot type:        increment (#2)
    Version name:         MTXSR00001086_00000002_00000009
    Version time:         10
    Publisher namespace:  xsrExample
    Subscriber namespace: xsrExample
```

When the last increment is loaded, the Subscriber database can de-establish the replication via the `mt_xsr unsubscribe` command and become the master database.

```
$ mt_xsr -d replica@localhost unsubscribe  -n xsrExample

$ mt_xsr -d replica@localhost describe --subscriber
No XML-based Snapshot Replication subscriber on database replica at
    time 14
```

mt_xsr publish

The `mt_xsr` utility with the `publish` command allows you to publish into XML documents the database incremental changes.

```
$ mt_xsr publish -h
MATISSE XML-based Snapshot Replication Manager x64 Version 9.1.0.0 (64-bit Edition) -
Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_xsr [OPTIONS] publish -f <xmlfile> [-s <size>[M|G]] [-p <n>] [-x <n>] [-n
<nsname>] [-d|-m] -a|-i [-h]
    -f, --file=...     Specify the XML-based Snapshot Replication document file.
                      The XML data is published into a collection of XML
                      segment files named <xmlfile>_xsr_do<docid>.xml,
                      <xmlfile>_xds_ia<docid>.xml, <xmlfile>_xsr_ir<docid>.xml,
                      <xmlfile>_xds_ua<docid>.xml and
                      <xmlfile>_xsr_ur<docid>.xml.
    -s, --size=...     Specify the XML segment file max size.
    -p, --parallel=... Publish data with <n> tasks running in parallel.
    -x, --prefetch=... Specify the number of objects to be prefetched when
                      exporting data. The default value is 128. The values
                      range between 1 and 128.
    -d, --iobuffer     Write XML data to the file in buffered I/O mode.
    -m, --iostream     Write XML data to the file in stream I/O mode.
    -a, --full         Publish the entire database.
    -i, --increment    Publish the database increment since the last
                      publication.
    -n, --ns=...       Specify the namespace from which the objects are
                      exported.
    -h, --help         Display this help and exit.
```

mt_xsr subscribe

The `mt_xsr` utility with the `subscribe` command allows you to establish replication with a master database and to synchronize with the master by loading the database incremental changes from XML documents.

```
$ mt_xsr subscribe -h
MATISSE XML-based Snapshot Replication Manager x64 Version 9.1.0.0 (64-bit Edition) -
Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_xsr [OPTIONS] subscribe -f <xmlfile> [-n <nsname>] [-p <n>] [-x <n>] [-c <n>] [-h]
    -f, --file=...     Specify the XML-based Snapshot Replication document file
                      to be loaded into the database.
    -n, --ns=...       Specify the subscriber namespace into which the objects
                      are imported. When the --ns option is ommitted, each
```

```
                    object is imported in a namespace matching the schema
                    class namespace.
    -p, --parallel=... Import data with multiple tasks running in parallel. The
                    number of tasks is limited by the number of XML segment
                    files.
    -x, --parse=...    Specify the number of xml objects to be parsed in one
                    sequence. The default value is 256 (1024 in parallel
                    mode). The values range between 1 and 2048.
    -h, --help         Display this help and exit.
```

mt_xsr describe

The `mt_xsr` utility with the `describe` command allows you to view publishers and subscribers settings information.

```
$ mt_xsr describe -h
MATISSE XML-based Snapshot Replication Manager x64 Version 9.1.0.0 (64-bit Edition) -
Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_xsr [OPTIONS] describe [-a|-p|-s] [-f <xml_file>] [-h]
    -a, --all        Provide publishers and subscribers settings information
                    from the database.
    -p, --publisher  Provide publishers settings information from the database.
    -s, --subscriber Provide subscribers settings information from the database.
    -f, --file=...   Specify the XML-based Snapshot Replication document file
                    to be checked.
    -h, --help       Display this help and exit.
```

mt_xsr unpublish

The `mt_xsr` utility with the `unpublish` command allows you to de-establish the replication of a the master database with a replica database.

```
$ mt_xsr unpublish -h
MATISSE XML-based Snapshot Replication Manager x64 Version 9.1.0.0 (64-bit Edition) -
Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_xsr [OPTIONS] unpublish -a | -n <nsname> [-h]
    -a, --all     Remove all publishers settings from the database.
    -n, --ns=...  Specify the namespace in the database from which the publisher
                 settings are removed.
    -h, --help    Display this help and exit.
```

mt_xsr unsubscribe

The `mt_xsr` utility with the `unsubscribe` command allows you to de-establish the replication of a replica database with a master database.

```
$ mt_xsr unsubscribe -h
MATISSE XML-based Snapshot Replication Manager x64 Version 9.1.0.0 (64-bit Edition) -
Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_xsr [OPTIONS] unsubscribe -a | -n <nsname> [-h]
    -a, --all     Remove all subscribers settings from the database.
    -n, --ns=...  Specify the subscriber namespace in the database from which
```

```
                the subscriber settings are removed.
-h, --help      Display this help and exit.
```

1.3 Database Activity Monitoring

Matisse new database activity monitoring capability helps database administrators monitor the performance and availability of databases on distributed network. It provides key performance metrics for ensuring the database server runs efficiently.

Performance graphs are available instantly in the Enterprise Manager Monitoring Panel providing insight into the performance over a period of time. It delivers both historical and current performance metrics on CPU and Disk I/O usage, on server cache hit ratio as well as on connection and request volume.

In addition the `mt_server logactivity` command manages the recording of historical performance metrics and generates performance reports.

1.4 Direct-io datafiles

Data files can be file system files or raw partitions. Raw partitions offer better performance and reliability, but in most cases it is more complex to manage. Direct-io datafiles provide the performance and reliability that we normally see only in raw devices, but also hides the complexity of managing raw devices, while still letting DBAs directly deal with files, bringing data file management to the next level of simplicity, performance and reliability.

The `DATIODIRECT` optional database configuration parameter has been added to manage direct-io datafiles.

1.5 Transparent IPv6 support

Matisse servers and services can transparently accept both `IPv4` and `IPv6` connections. The commands that require to specify a hostname supports both the `IPv4` address format (i.e. `192.168.1.87`) and the `IPv6` address format (i.e `fd01:192:168:1::87`).

Most tools and utilities define the database source name with the `<dbname>[@<hostname>[:<port>]]` format. The alternative syntax for an `IPv6` address is `<hostname>[/<port>]` or by enclosing the `IP` address in square brackets (i.e. `mydb@[fd01:192:168:1::87]`).

1.6 Enterprise Manager Tool

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

Database Monitoring

The Enterprise Manager Monitoring Panel includes the Activity and History panels to deliver graphs on respectively current and historical performance metrics.

Import/Export XML

The Import and Export XML data dialogs have been updated to add new options in the Advanced Options panel. Progress information of the processing is provided during the XML operation.

Database Properties

The Create and Update Database dialogs manage the `DATIODIRECT`, `AUTOCOLLECTFREQ`, `OBJTABLESIZ`, `OBJTABCLRFREQ` and `OBJTABCLRLEVEL` configuration parameters.

1.7 Matisse SQL

The new Matisse SQL features enhance ad-hoc reporting capabilities.

DDL create

SQL DDL `CREATE` statements supports the existence check clause `[IF [NOT] EXISTS [schema_object]]` providing more flexible ways to manage schema upgrade with SQL statements.

```
CREATE NAMESPACE [IF [NOT] EXISTS [schema_object]]
    <nsname>.<subnsname>

CREATE {CLASS | TABLE} [IF [NOT] EXISTS [schema_object]] <class>
      [ {UNDER | INHERIT} <superclass> [, ...] ]

CREATE [UNIQUE] INDEX [IF [NOT] EXISTS [schema_object]] <index>
      ON <class>

CREATE [UNIQUE] ENTRY_POINT DICTIONARY [IF [NOT] EXISTS
    [schema_object]] <ep_dict>
      ON <class>

CREATE [INSTANCE | STATIC] METHOD [IF [NOT] EXISTS [schema_object]]
    <method name>
        ( <parameter declaration> [, ... ] )
        RETURNS <data type>
        FOR <class name>

With schema_object such as:

schema_object ::=
 SCHEMA_OBJECT(NAMESPACE,[<ns path name>.]<ns name>)
 | SCHEMA_OBJECT(CLASS,[<ns path name>.]<class name>)
 | SCHEMA_OBJECT(ATTRIBUTE,[<ns path name>.]<cls name>.<att name>)
 | SCHEMA_OBJECT(RELATIONSHIP,[<ns path name>.]<cls name>.<rel name>)
 | SCHEMA_OBJECT(METHOD,[<ns path name>.]<cls name>.<mth name>)
 | SCHEMA_OBJECT(INDEX,[<ns path name>.]<index name>)
 | SCHEMA_OBJECT(ENTRY_POINT DICTIONARY,[<ns path name>.]<entry point
    name>)
```

DDL alter

SQL DDL `ALTER` statements supports the existence check clause `[IF [NOT] EXISTS [schema_object]]` providing more flexible ways to manage schema upgrade with SQL statements.

```
ALTER NAMESPACE [IF [NOT] EXISTS [schema_object]]
    <nsname>[.<subnsname>] RENAME TO <new_nsname>

ALTER {CLASS | TABLE} [IF [NOT] EXISTS [schema_object]] <class>
          DROP { ATTRIBUTE <attribute>
               | RELATIONSHIP <relationship>
               | {INHERIT | UNDER} <superclass>
               }

       ALTER {CLASS | TABLE} [IF [NOT] EXISTS [schema_object]] <class>
          ADD { ATTRIBUTE <attribute> <attribute_type>
                  [DEFAULT <literal>] [NOT NULL]
              | RELATIONSHIP <relationship>
                  [[READONLY] REFERENCES [SET | LIST]]
                    ( <succ_class> [, ...] )
                  [CARDINALITY (<min>, <max>)]
                  [INVERSE <inv_class>.<inverse_rshp>]
              | {INHERIT | UNDER} <superclass>
              }

       ALTER {CLASS | TABLE} [IF [NOT] EXISTS [schema_object]] <class>
          ALTER { ATTRIBUTE <attribute> <attribute_type>
                    [DEFAULT <literal>] [NOT NULL]
                | RELATIONSHIP <relationship>
                    [[READONLY] REFERENCES [SET | LIST]]
                      ( <succ_class> [, ...] )
                    [CARDINALITY (<min>, <max>)]
                    [INVERSE <inv_class>.<inverse_rshp>]
                }

       ALTER {CLASS | TABLE} [IF [NOT] EXISTS [schema_object]] <class>
          RENAME { TO <new_class_name>
                 | ATTRIBUTE <attribute> TO <new_attribute_name>
                 | RELATIONSHIP <relationship> TO
    <new_relationship_name>
                 }

With schema_object such as:

schema_object ::=
 SCHEMA_OBJECT(NAMESPACE,[<ns path name>.]<ns name>)
 | SCHEMA_OBJECT(CLASS,[<ns path name>.]<class name>)
 | SCHEMA_OBJECT(ATTRIBUTE,[<ns path name>.]<cls name>.<att name>)
 | SCHEMA_OBJECT(RELATIONSHIP,[<ns path name>.]<cls name>.<rel name>)
 | SCHEMA_OBJECT(METHOD,[<ns path name>.]<cls name>.<mth name>)
 | SCHEMA_OBJECT(INDEX,[<ns path name>.]<index name>)
 | SCHEMA_OBJECT(ENTRY_POINT DICTIONARY,[<ns path name>.]<entry point
    name>)
```

DDL drop

SQL DDL `DROP` statements supports the existence check clause `[IF [NOT] EXISTS [schema_object]]` providing more flexible ways to manage schema upgrade with SQL statements.

```
DROP NAMESPACE [IF [NOT] EXISTS [schema_object]
    <nsname>[.<subnsname>]

DROP TABLE [IF [NOT] EXISTS [schema_object] tbl_name

DROP INDEX [IF [NOT] EXISTS [schema_object]] index_name ON tbl_name

DROP ENTRY_POINT DICTIONARY [IF [NOT] EXISTS [schema_object]]
    <ep_dictionary>

DROP METHOD [IF [NOT] EXISTS [schema_object]] <method name> FOR <class
    name>

With schema_object such as:

schema_object ::=
 SCHEMA_OBJECT(NAMESPACE,[<ns path name>.]<ns name>)
 | SCHEMA_OBJECT(CLASS,[<ns path name>.]<class name>)
 | SCHEMA_OBJECT(ATTRIBUTE,[<ns path name>.]<cls name>.<att name>)
 | SCHEMA_OBJECT(RELATIONSHIP,[<ns path name>.]<cls name>.<rel name>)
 | SCHEMA_OBJECT(METHOD,[<ns path name>.]<cls name>.<mth name>)
 | SCHEMA_OBJECT(INDEX,[<ns path name>.]<index name>)
 | SCHEMA_OBJECT(ENTRY_POINT DICTIONARY,[<ns path name>.]<entry point
    name>)
```

Coalesce

`Coalesce` evaluates the arguments in order and returns the current value of the first expression that initially does not evaluate to `NULL`. If all arguments are `NULL`, `COALESCE` returns `NULL`.

```
SELECT a.firstName, a.lastName,
       COALESCE(a.firstName, a.lastName) AS firstNotNull
FROM artist AS a;
```

Nullif

`Nullif` returns the first expression if the two expressions are not equal. If the expressions are equal, `NULLIF` returns a null value of the type of the first expression.

```
SELECT AVG(NULLIF(runningTime, 0)) FROM movie;
```

1.8 Schema Manager

Matisse Schema Manager utility (`mt_sdl`) has been updated to be consistent with the other commands which support short and long name options as well as database source name compatible with IPv6 address.

mt_sdl import

The `mt_sdl` utility with the `import` command allows you to import the database schema in ODL or SQL DDL format.

```
$ mt_sdl import -h
MATISSE Schema Definition Language x64 Version 9.1.0.0 (64-bit Edition) - Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.
```

```
Usage:
  mt_sdl [OPTIONS] import -f <schema file> {-o | -d} [-h]
    -f, --file=... Specify the schema definitions script file to be loaded
                   into the database.
    -o, --odl      Load the ODL class definitions into the database.
    -d, --ddl      Load the SQL DDL script into the database.
    -h, --help     Display this help and exit.
```

mt_sdl export

The `mt_sdl` utility with the `export` command allows you to export the database schema in ODL or SQL DDL format.

```
$ mt_sdl export -h
MATISSE Schema Definition Language x64 Version 9.1.0.0 (64-bit Edition) - Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_sdl [OPTIONS] export -f <schema file> {-o | -d} [-n <namespace>] [-h]
    -f, --file=...  Specify the schema definitions script file to be generated
                    from the database.
    -o, --odl       Generate the ODL class definitions from a database schema.
    -d, --ddl       Generate the SQL DDL script from a database schema.
    -n, --ns=...    Export only the schema objects under the provided namespace
    -h, --help      Display this help and exit.
```

mt_sdl parse

The `mt_sdl` utility with the `parse` command allows you to parse for conformance the database schema in ODL or SQL DDL format.

```
$ mt_sdl parse -h
MATISSE Schema Definition Language x64 Version 9.1.0.0 (64-bit Edition) - Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_sdl parse -f <schema file> {-o | -d} [-h]
    -f, --file=...  Specify the schema definitions script file to be parsed.
    -o, --odl       Parse the ODL class definitions.
    -d, --ddl       Parse the SQL DDL script.
    -h, --help      Display this help and exit.
```

mt_sdl stubgen

The `mt_sdl` utility with the `stubgen` command allows you to generate source code from the database schema classes defined in the ODL file.

```
$ mt_sdl stubgen -h
MATISSE Schema Definition Language x64 Version 9.1.0.0 (64-bit Edition) - Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_sdl stubgen { -l cxx [-s <namespace>] [-n <namespace>]
                 | -l java [-s <namespace>] [-n <package>] [-m]
                 | -l php [-s <namespace>] [-n <namespace>]
                 | -l python [-s <namespace>]
                 | -l eiffel [-s <namespace>] } -f <ODL file> [-h]
    -f, --file=...    Specify the ODL class definitions file.
    -l, --lang cxx    Create C++ files from the ODL class definitions.
    -l, --lang java   Create Java files from the ODL class definitions.
    -l, --lang php    Create PHP files from the ODL class definitions.
    -l, --lang python Create Python files from the ODL class definitions.
```

```
    -l, --lang eiffel Create Eiffel files from the ODL class definitions.
    -s, --sn=...      Specify the schema class namespace that is mapped to a
                      language class namespace if any and if language supports
                      namespaces.
    -n, --ln=...      Specify the language class namespace for the generated
                      proxi classes. when the --sn and --ln options are omitted,
                      each class is generated in a namespace matching the schema
                      class namespace.
    -m, --psm         Generate methods mapping SQL method calls.
    -h, --help        Display this help and exit.
```

1.9 Data Transformation Services

Matisse Data Transformation Services utility (`mt_dts`) has been updated to be consistent with the other commands which support short and long name options as well as database source name compatible with IPv6 address.

mt_dts import

The `mt_dts` utility with the `import` command allows you to load data in a `CSV` format into the database server.

```
$ mt_dts import -h
MATISSE Data Transformation Services x64 Version 9.1.0.0 (64-bit Edition) - Apr 29
2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_dts [OPTIONS] import -f <CSV file> [-c <class name>] [-u] [-n] [-m File|Column]
    -f, --file=...  Specify the CSV file to be loaded.
    -c, --class=... Specify the class name where the data will be loaded if
                    different from the CSV filename.
    -u, --update    When specified with the first columns of the CVS file
                    composing the primary key, values of existing objects are
                    updated.
    -n, --noname    When specified there is no field name on the first row in
                    the CVS file.
    -m, --media=... When specified with 'File', the media data is in an external
                    file. With 'Column' the media data is in the CVS file.
```

mt_dts export

The `mt_dts` utility with the `export` command allows you to extract data in a `CSV` format from the database server.

```
$ mt_dts export -h
MATISSE Data Transformation Services x64 Version 9.1.0.0 (64-bit Edition) - Apr 29
2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_dts [OPTIONS] export -f <CSV file> [-q "<SQL select>" | -c <class name>] [-n] [-
m File|Column]
    -f, --file=...  Specify the CSV file to be generated.
    -q, --sql=...   Specify the SQL select statement which filters data to be
                    exported.
    -c, --class=... Specify the class containing the data to be exported.
    -n, --noname    When specified there is no field name on the first row in
```

```
                the CVS file.
    -m, --media=... When specified with 'File', the media data is exported into
                an external file. With 'Column' the media data is exported
                into the CVS file.
```

mt_dts link

The `mt_dts` utility with the `link` command allows you to establish relationships between objects as described in the `XRD` file.

```
$ mt_dts link -h
MATISSE Data Transformation Services x64 Version 9.1.0.0 (64-bit Edition) - Apr 29
2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_dts [OPTIONS] link -f <XRD file>
    -f, --file=... Specify the XML Relationship Descriptors file describing how
                to establish relationship between entities.
```

1.10 XML Manager

Matisse XML has been extended to fully cover update and delete operations with XML documents.

Matisse XML Manager utility (`mt_xml`) has been updated to be consistent with the other commands which support short and long name options as well as database source name compatible with IPv6 address.

Updates

The XML attribute `MtAction="update"` indicates that the object is to be updated. The object is accessed by its primary key and each property value defined in the XML object is updated accordingly.

```
<?mt_xml version="2"
       container="yes"
       oid="yes"
       prealloc="1"?>
<MtContainer>

<xmlExample.Category oid="4255" MtAction="update">
<name MtBasicType="MT_STRING">Category 02 updated</name>
</xmlExample.Category>

</MtContainer>
```

Deletes

The XML attribute `MtAction="delete"` indicates that the object is to be deleted. The object is accessed by its primary key.

```
<?mt_xml version="2"
       container="yes"
       oid="yes"
       prealloc="2"?>
<MtContainer>

<xmlExample.Category oid="4252" MtAction="delete"/>
```

```
<xmlExample.Document oid="4249" MtAction="delete"/>

</MtContainer>
```

mt_xml import

The `mt_xml` utility with the `import` command allows you to load XML documents into the database.

```
$ mt_xml import -h
MATISSE XML Manager x64 Version 9.1.9.0 (64-bit Edition) - Jun 29 2016.
(c) Copyright 2016 Matisse Software Inc. All rights reserved.MATISSE XML Manager x64

Usage:
  mt_xml [OPTIONS] import {-f <xmlfile> | -i} [-n <nsname>] [-d <nsname>] [-p <n>] [-
x <n>] [-u] [-c <n> | -o] [-h]
    -f, --file=...     Specify the XML data file to be loaded into the database.
    -i, --in           Read the XML data from standard input and import it into
                       the database.
    -n, --fn=...       Specify the namespace from which the objects are
                       imported.
    -d, --dn=...       Specify the namespace into which the objects are
                       imported. When the --fn and --dn options are ommitted,
                       each object is imported in a namespace matching the
                       schema class namespace.
    -p, --parallel=... Import data with <n> tasks running in parallel. The XML
                       data is imported from a multi-segment XML file. The
                       number of tasks is limited by the number of XML file
                       segments.
    -x, --parse=...    Specify the number of xml objects to be parsed in one
                       sequence. The default value is 256 (1024 in parallel
                       mode). The values range between 1 and 2048.
    -u, --update       When specified with MtPrimaryKey attribute, values of
                       existing objects are updated.
    -c, --commit=...   Commit transaction for every <n> objects created. (by
                       default commit occurs every 20480 objects created)
    -o, --onecommit    Forces to import the XML document in a single
                       transaction (require enough memory to parse the XML
                       document and to create all the objects in memory)
    -h, --help         Display this help and exit.
```

mt_xml export

The `mt_xml` utility with the `export` command allows you to export XML documents from the database.

```
$ mt_xml export -h
MATISSE XML Manager x64 Version 9.1.9.0 (64-bit Edition) - Jun 29 2016.
(c) Copyright 2016 Matisse Software Inc. All rights reserved.

Usage:
  mt_xml [OPTIONS] export {-f <xmlfile> [-s <size>[M|G]] | -o} [-e] [-k] [-p <n> | -t
<n>] [-l <n>] [-x <n>] [-n <nsname
>] [-d|-m] {-a | -q <stmt> | -i <oid>} [-h]
    -f, --file=...     Specify the XML data file storing XML data extracted from
                       the database.
    -s, --size=...     Specify the XML data file max size therefore splitting
                       XML data into multiple XML files named
                       <xmlfile>_xds_<docid>.xml.
    -o, --out          Write XML data to the standard output.
```

```
-e, --emedia[=...] Export media data in the XML document instead of
                   exporting media data into external files. When specified
                   the value indicates the maximum media data size in bytes
                   to be exported in the XML document. Media data above this
                   threshold are exported into external files.
-k, --pkoid         Export data in a format with OIDs in the xml tags to
                   enable Primary Key recovery. (The --full option always
                   exports in this format)
-t, --task=...      Export data with <n> tasks running in parallel. The XML
                   data is exported into multiple XML files named
                   <xmlfile>_xds_<docid>.xml.
-p, --parallel=... Export data with <n> tasks running in parallel. The XML
                   data is exported into multiple XML files named
                   <xmlfile>_xds_a<docid>.xml and <xmlfile>_xds_r<docid>.xml
                   This is the XML format to import a multi-segment XML
                   file in parallel.
-l, --split=...     Specify the number of objects in a class which triggers a
                   object split per task in parallel export. The minimum
                   value is 100.
-x, --prefetch=... Specify the number of objects to be prefetched when
                   exporting data. The default value is 128. The values
                   range between  1 and 128.
-d, --iobuffer      Write XML data to the file in buffered I/O mode.
-m, --iostream      Write XML data to the file in stream I/O mode.
-a, --full          Export all non-schema data into one or multiple XML
                   files.
-q, --sql=...       Specify the SQL SELECT statement retrieving the objects
                   to be exported.
-i, --oid=...       Specify the list of object OIDs to be exported. Both
                   decimal and hexadecimal oid formats are accepted.
-n, --ns=...        Specify the namespace from which the objects are
                   exported.
-h, --help          Display this help and exit.
```

mt_xml parse

The `mt_xml` utility with the `parse` command allows you to parse XML documents to check the conformance of the document.

```
]$ mt_xml parse -h
MATISSE XML Manager x64 Version 9.1.0.0 (64-bit Edition) - Apr 29 2013.
(c) Copyright 2013 Matisse Software Inc. All rights reserved.

Usage:
  mt_xml parse {-f <xml_file> | -i} [-x <n>] [-t] [-h]
    -f, --file=... Specify the XML data file to be parsed.
    -i, --in       Read the XML data from standard input and parse it.
    -x, --parse     Specify the number of objects to be parsed in one data
                   sequence.
                   The default value is 256. The values range between 1 and
                   2048.
    -t, --info      Reports the progress during the parsing.
    -h, --help      Display this help and exit.
```

Parallel Import

The following example imports the XML documents in the database with 6 tasks running in parallel:

```
$ mt_xml -d example import -f outp6/exampleP6.xml -parallel 6
```

Verbosity Level

With verbose level set to 1, XML commands show statistics information when the execution is completed.

```
$ mt_xml -d example --verbose=1 import -f reportsP2.xml --parallel 2
  [STAT] Number of top-level xml objects read: 30065
  [STAT] Number of objects created: 30065
  [STAT] Size of oid mapping table: 0.48 MB
  [OPTN] Number of tasks: 2
  [OPTN] Number of xml objects parsed at once: 1024
  [STAT] Number of deadlocks: 0
```

With verbose level set to 2, XML commands show execution progress in addition to providing statistics information.

```
$ mt_xml -d example --verbose=2 import -f reportsP2.xml --parallel 2
  [INFO] task #1 loading reportsP2_xds_a001.xml
  [INFO] task #2 loading reportsP2_xds_a003.xml
  [INFO] task #1 loading reportsP2_xds_r002.xml
  [INFO] task #2 loading reportsP2_xds_r004.xml
  [STAT] Number of top-level xml objects read: 30065
  [STAT] Number of objects created: 30065
  [STAT] Size of oid mapping table: 0.48 MB
  [OPTN] Number of tasks: 2
  [OPTN] Number of xml objects parsed at once: 1024
  [STAT] Number of deadlocks: 0
  [TIME] Start loading:                    17:35:34.432    Elapsed
    00:00:00.000
  [TIME] Attr loaded  :                    17:35:36.205    Elapsed
    00:00:01.773
  [TIME] Rshp loaded  :                    17:35:37.645    Elapsed
    00:00:01.439
  [TIME] Total loading:                    17:35:37.645    Elapsed
    00:00:03.212
```

Export I/O Mode

The XML export command with the `--iobuffer` option writes XML documents in buffered I/O mode while the `--iostream` option writes XML documents in stream I/O mode. On Windows for example the buffered I/O mode is significantly faster than the stream I/O mode.

```
$ mt_xml -d example --verbose=2 export -f out/reportsP2.xml --full --
    parallel 2 --iobuffer
  [INFO] task #1 writing out/reportsP2_xds_a001.xml
  [INFO] task #1 writing out/reportsP2_xds_r002.xml
  [INFO] task #2 writing out/reportsP2_xds_a003.xml
  [INFO] task #2 writing out/reportsP2_xds_r004.xml
  [STAT] Number of top-level objects exported: 90205
  [OPTN] Number of prefetch objects: 128
  [OPTN] XML data with OID xml attribute: YES
  [OPTN] Media data into external files: YES
  [OPTN] XML data file I/O mode: buffered
  [OPTN] Number of tasks: 2
  [TIME] Start schema info building:       17:51:02.065    Elapsed
    00:00:00.000
  [TIME] End schema info building  :       17:51:02.066    Elapsed
    00:00:00.001
  [TIME] Start extracting:                 17:51:02.016    Elapsed
    00:00:00.000
```

```
[TIME] End extracting  :                17:51:03.589    Elapsed
  00:00:01.573
```

1.11 Java Binding

The Java binding has been upgraded to support Java 8.

Java 8

Matisse Java binding has been upgraded to support Java 8.

1.12 Database Utility Commands

The utility commands trace more execution information as the verbosity level increases.

mt_server logactivity

The `mt_server logactivity` command allows you to manage the recordings of the database activity metrics.

```
$ mt_server logactivity -h
Usage: mt_server [OPTIONS] logactivity -s | -e | -w -f <csv file> -r
    h6|h12|d1|d7|d30 [-h]
  -s, --start     Start the recording of the database activity
  -e, --end       End the recording of the database activity session
  -i, --info      Provide information about the database recording
    activity
  -w, --write     Write the database activity into a CSV file
  -f, --file=...  Specify the CSV file
  -r, --range=... Time range elected from the database activity
    session (default h6)
  -h, --help      Display this help and exit
```

mt_server setrunfrequency

The `mt_server setrunfrequency` command allows you to set the run frequency of automatic operations.

```
$ mt_server setrunfrequency -h
Usage: 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
```

mt_server setrunlevel

The `mt_server setrunlevel` command allows you to set the run level of automatic operations.

```
$ mt_server setrunlevel -h
Usage: mt_server [OPTIONS] setrunlevel [-c] -l <level>
  -c, --clearot   Set clear object table run level
  -l, --level=... run level in percentage from 10 to 90 (0 for
    automatic)
  -h, --help      Display this help and exit
```

mt_server info

The `mt_server info` command provides detailled information about an online database.

```
$ mt_server info -h
Usage: mt_server [OPTIONS] info [-Vlbcpsth]
  -a, --all          List all information (default)
  -V, --version      Server version
  -l, --ltime        Current logical time
  -b, --backup-ltime  Last backup logical time
  -c, --collect-ltime Highest collected logical time
  -p, --page-cache    Page cache size (in Mbytes)
  -s, --surrogate     Max surrogate
  -t, --tran-mgr      Transaction manager state
  -h, --help         Display this help and exit
```

mt_transaction log

The `mt_transaction log` command enables/disables the transaction deadlock information logging.

```
$ mt_transaction log -h
Usage: mt_transaction [OPTIONS] log [-h]
  -e, --enable    Enable logging
  -d, --disable   Disable logging (default)
  -h, --help      Display this help and exit
```

mt_ios schema

The `mt_ios schema` command provides information about schema internal objects.

```
$ mt_ois schema -h
Usage: mt_ois [OPTIONS] schema [-h] [-o <oid>] [-s <size]
  -o, --oid=...   Internal Object id
  -s, --size=...  Display size for string types (default 24
    characters)
  -h, --help      Display this help and exit
```

mt_ios class

The `mt_ios class` command provides information about a class internal objects.

```
$ mt_ois class -h
Usage: mt_ois [OPTIONS] class [-n <name> | -c <cid>] [-o <oid>] [-s
    <size] [-h]
  -n, --name=...  Full class name
  -c, --cid=...   Class object id
  -o, --oid=...   Internal Object id
  -s, --size=...  Display size for string types (default 24
    characters)
  -h, --help      Display this help and exit
```

mt_ios index

The `mt_ios index` command provides information about an index internal objects.

```
$ mt_ois index -h
Usage: mt_ois [OPTIONS] index [-n <name> | -i <idx>] [-o <oid>] [-s
    <size] [-h]
  -n, --name=...  Full index name
  -i, --idx=...   Index object id
  -o, --oid=...   Internal Object id
```

```
  -s, --size=...  Display size for string types (default 24
    characters)
  -h, --help      Display this help and exit
```

mt_ios dictionary

The `mt_ios dictionary` command provides information about a dictionary internal objects.

```
$ mt_ois dictionary -h
Usage: mt_ois [OPTIONS] dictionary [-n <name> | -d <dict>] [-o <oid>]
    [-s <size] [-h]
  -n, --name=...  Full dictionary name
  -d, --dict=...  Dictionary object id
  -o, --oid=...   Internal Object id
  -s, --size=...  Display size for string types (default 24
    characters)
  -h, --help      Display this help and exit
```

mt_ios object

The `mt_ios object` command provides information about an object internal structure.

```
$ mt_ois object -h
Usage: mt_ois [OPTIONS] object -o <oid> [-h]
  -o, --oid=...   Object id
  -h, --help      Display this help and exit
```

1.13 Database Configuration

A set of new optional database configuration parameters have been added to fine tune the performance of production databases.

DATIODIRECT

This parameter defines the I/O mechanism for data files on the file system. When set to 1, it minimizes the file system cache effects of the I/O to and from the data file.

AUTOCOLLECT FREQ

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 value 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 value is 120 seconds.

OBJTABCLRLEVEL

This parameter defines the minimum level of space used in the multi-version object table memory cache before a clearing operation is triggered. This parameter is expressed in percentage. The minimum value is 10%, the maximum value is 90%. Specifying a value of 0 disables the control of the object table clearing operation by the parameter. The automatic version collection is clearing the object table memory cache.

TCPKEEPALIVE

This parameter defines the keepalive control of the server TCP/IP connections. When enabled, it verifies on a regular basis that the endpoint at the remote end of the connection is still available. OS-specific value of the keepalive intervals are controllable at the system level. Specifying a value of 1 enables keepalive control. Specifying a value of 0 disables keepalive control.

On Windows, the default settings when a TCP socket is initialized sets the keep-alive time-out to 2 hours and the keep-alive interval to 1 second. The default system-wide value of the keep-alive time-out is controllable through the KeepAliveTime registry setting which takes a value in milliseconds. The default system-wide value of the keep-alive interval is controllable through the KeepAliveInterval registry setting which takes a value in milliseconds. The number of keep-alive probes (data retransmissions) is set to 10 and cannot be changed.

On Linux, the default settings when a TCP socket is initialized sets the keep-alive time-out to 2 hours and the keep-alive interval to 75 second. The number of keep-alive probes (data retransmissions) is set to 9. The default system-wide value of the keep-alive parameters is controllable through the following files:
/proc/sys/net/ipv4/tcp_keepalive_intvl
/proc/sys/net/ipv4/tcp_keepalive_probes
/proc/sys/net/ipv4/tcp_keepalive_time

1.14 License Key Format

The customer license key format has changed in release 9.1. Matisse 9.1 does not recognize license keys issued for prior releases. Upon installation of Matisse 9.1, a license key with limited features is automatically issued.

2 Compatibility with Previous Releases

2.1 Matisse 9.1 Data Migration

Matisse Server 9.1 comes with several changes in the data format. You must use the `mt_xml` or the `mt_xsr` tool to convert an existing database (9.0.x or prior) into the 9.1 format.

Step 1

Before installing 9.1.x, if you convert an existing database (9.0.x or prior) you need to check if your data and your application are compatible with Matisse 9.1 since the array datatypes are no longer public. Most likely you are already using the list datatypes since the vast majority of the Matisse language bindings do not support Matisse array datatypes. To double-check, run the following SQL statement, which does not return any row if your application is compatible.

```
$ mt_sql -d <dbname>
SQL>SELECT
MtBasicType,
MtAttributeTypeOf.MtName AS AttributeName,
MtAttributeTypeOf.MtAttributeOf.MtName AS ClassName
FROM MtType
WHERE MtBasicType IN LIST(INT)(16, 20, 24, 40, 44, 48);
```

However if the query returns one or more rows, you need to migrate your data and upgrade your application. Replace the Array data type with the corresponding List data type.

Step 2

Before installing 9.1.x, save your schema in ODL and your data in XML format. To minimize the downtime of your application we recommend that you evaluate the parallel option for XML export.

To backup your schema in ODL:

```
$ mt_sdl -d <dbname> export --odl -f schema.odl
```

To backup your data in XML:

```
$ mt_xml -d <dbname> export -f data.xml -s <m>G --full
```

You may check the *Matisse XML Programming Guide* for more options to export in XML format.

Step 3

You may now install Matisse 9.1.x on your machine and then restore the schema and the data as follows:

```
$ mt_sdl -d <dbname> import --odl -f schema.odl
```

```
$ mt_xml -d <dbname> import -f data.xml --parallel <n>
```

You may check the *Matisse XML Programming Guide* for more options to import in XML format.

2.2 Client Connections

Only 9.1.x clients may be used with 9.1.x servers.

The clients for earlier releases of Matisse are incompatible with the 9.1.x server. Consequently, you must upgrade any older clients to 9.1.x before attempting to access a 9.1.x server.

3 Platform-Specific Topics

3.1 Linux

The most popular Linux distributions on x86 (32-bit) and x86_64 (64-bit) chip families are supported. Any Linux distribution, where Matisse DBMS has not been tested, require Linux kernel 2.6.18 or higher on systems based on x86 (32-bit) or x86_64 (64-bit) chip families.

3.2 MacOS

Support for MacOS X version for Intel (10.5 Sierra or higher).

3.3 Solaris

Support for Solaris 11 on x86 (32-bit) and x86_64 (64-bit) chip families.

The Solaris 11 on SPARC with 32-bit kernel and 64-bit kernel is available upon request.

3.4 Windows

Support for Windows (2008/2012/2016/Vista/7/8/8.1/10) on systems based on x86 (32-bit) and x86_64 (64-bit) chip 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 9.1.12

- The `mt_ois schema` command lists schema internal object statistics.
- The `mt_ois class` command lists class internal object statistics.
- The `mt_ois index` command lists index internal object statistics.
- The `mt_ois dictionary` command lists dictionary internal object statistics.
- The `mt_ois object` command shows internal object information.
- The `mt_connection monitor` command now monitors active database connections.

```
-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)
```

- In the Enterprise Manager Schema Panel, there are now menu-items to generate DDL statement scripts to create or drop an existing schema object.
- In the Enterprise Manager Schema Panel, the `EXIST` keyword is not highlighted as a reserved keyword.
- In the Enterprise Manager Schema Panel, SQL help for `CREATE INDEX` does not show the optional keyword `[NOT]` to qualify `UNIQUE` or `CASE SENSITIVE`.
- XML Parallel export now records statistics about each exported segment file and its content, document attributes includes `elapsed, objectCount, successorCount, mediaCount, mediaSize` and `fileSize`.

```
<MtDocument segment="2" filename="media91out_xds_a0002.xml"
elapsed="00:00:00.001573" fileSize="6673" objectCount="16" mediaCount="6"
mediaSize="35817"/>
<MtDocument segment="3" filename="media91out_xds_r0003.xml"
elapsed="00:00:00.001552" fileSize="1419" objectCount="16" successorCount="12"/>
```

Resolved in Matisse 9.1.11

- The `--split` option of `mt_sdl export` command now exports all `SQL` statements by statement type (table, index, method) in the generated `SQL DDL` file.

```
-s, --split     Split all SQL statements by type in the SQL DDL script.
```

- The `mt_sdl import` command may fail with error `MATISSE-E-INVALID_STMTATT` if the index descriptors in the `ODL` file were removed from the `ODL` class descriptor defining `SQL` methods referencing any one of these indexes.

- The `mt_sdl import` command may fail with error `MATISSE-E-OBJECTNOTFOUND` if the ODL class descriptor adds on a class with instances an attribute and an index using this attribute.

- A `SQL SELECT` statement may fail to compile when the statement uses an Entry Point dictionary defined in a namespace.

```
sql> select * from test.Patient where entry_point(test.PatientDict) = 'abc';
>>> Error at line 1, position 36: 8648912 [MATISSE-E-SYNTAX_ERROR, MATISSE SQL
syntax error]
```

- The database log file now reports detailed information about the index maximum key size.

```
17 Mar. 2018 18:24:27 MIR-I-IDXKEYSIZ, Index: maximum key size set to 256 bytes
17 Mar. 2018 18:24:27 MIR-I-DICTKEYSIZ, Dictionary: maximum key size set to 256
bytes
```

- The `mt_datafile analyze` command now includes new options to further analyze a datafile content.

```
-D, --deleted       Print out deleted objects
-l, --low=...       Lowest ojbid to analyze
-i, --high=...      Highest ojbid to analyze
```

Resolved in Matisse 9.1.10

- The `mt_server info` command now provides the object id surrogate value.

```
$ mt_server -d test2 info
[...]
Surrogate:
Internal: 4097
Attribute: 4097
Relationship: 4097
Transaction: 4096
```

- The `--surrogate` option of `mt_server info` command returns the max surrogate value of an online database.

```
-s, --surrogate     Max surrogate
```

- The `mt_server monitor` command now shows the I/O rate maximum value.

- The `--buffer` option of `mt_xml import` command sets the connection buffer size.

```
-b, --buffer=...    Modify buffer message size to 64, 128, 256 or 512 Kbytes
                    The default value is 128 Kbytes.
```

- The `--buffer` option of `mt_xml export` command sets the connection buffer size.

```
-b, --buffer=...    Modify buffer message size to 64, 128, 256 or 512 Kbytes
                    The default value is 128 Kbytes.
```

- The `--force` option of `mt_xml export` command sets a flag or combination of flags to alter the import process.

```
-z, --force=...     Flag 1: Do not create temporary version name for parallel mode
(unsafe).
                    Flag 2: Ignore errors and discard invalid objects (unsafe).
```

- The `mt_xml export` command fails to export in parallel returning `MATISSE_OIDEXHAUSTED` error if the database has already exhausted all the OIDs for internal objects.
- The `mt_xml import` command fails to import in parallel more than 256 segment files.

Resolved in Matisse 9.1.9

- In a specific case a Matisse program can return an error `MATISSE-E-OBJECTNOTFOUND, 0xe0a0f not found` while one successor in a relationship is not properly read.
- The `mt_xml export` command now exports the media files across 3-level directory tree (`MediaFiles/m<xyz>/f<xyz>/<file>`) to reduce the number of files per directory and to improve the export speed.
- The `--emedia` option of `mt_xml export` command embeds media data into the XML file when the size is less than the defined value.

```
-e, --emedia[=...] Export media data in the XML document instead of exporting
media data into external files. When specified the value indicates the maximum
media data size in bytes to be exported in the XML document. Media data above this
threshold are exported into external files.
```

- The `--split` option of `mt_xml export` command triggers a split of class instances into multiple tasks for parallel exports.

```
-l, --split=...    Specify the number of objects in a class which triggers a
object split per task in parallel export. The minimum value is 100.
```

- The `--sql` option of `mt_xml export` command may fail with a `Floating point exception` error when the exported objects contain media data to be exported in external files.
- In the Enterprise Manager, the restore of a backup button is always disabled regardless of the backup size or the number of backup files to be restored.
- The Java binding with Java 8 may not always release the memory allocated by the JNI layer.
- In the Java binding, the compilation of a `SQL SELECT` statement with numerous `OR` conditions may crash the program.

Resolved in Matisse 9.1.8

- The `mt_transaction log` command enables/disables the transaction deadlock information logging.
- The `mt_transaction monitor` command now provides transaction deadlock information.
- In the Java binding, the `MtDatabase.getTransactionId()` method returns the ID of the current active transaction.

```
public final int MtDatabase.getTransactionId()
```

- In the C++ binding, the `MtDatabase.getTransactionId()` method returns the ID of the current active transaction.

```
public MtOid MtDatabase.getTransactionId()
```

- In the .NET binding, the `MtDatabase.getTransactionId()` method returns the ID of the current active transaction.
  ```
  public int MtDatabase.getTransactionId()
  ```

Resolved in Matisse 9.1.7

- SQL DDL statements miss a control of existence od the schema objects in `CREATE`, `ALTER` and `DROP` statements.
- Matisse Java binding has been upgraded to support Java 8.
- In some cases, the compilation code of SQL Methods defined in an ODL file may not include index maintenance instructions for the indexes defined in the same class as the method until `COMPILE ALL` is executed.
- XML import does not properly check the format of an OID value.
- XML parse or import do not always report the line number of the parsed or imported object when an error occurs.
- The XML import of a large XML file may fail returning the error `MATISSE-E-SYSTEMERROR, System error 0x8468182.`

Resolved in Matisse 9.1.6

- The new Matisse SQL function `Coalesce` and `Nullif` enhance ad-hoc reporting capabilities.
- In SQL, the `WHERE` clause including a `OR` statement may fail to filter the objects when the statement include a aggregate function computing data through the path of a relationship.
  ```
  SELECT count(*) FROM MyClass1 c
  WHERE
  (c.rel1.MyClass2 IS NULL and c.rel2.MyClass2 IS NOT NULL
  and ((c.myatt1 + SUM(c.rel2.MyClass2.myatt2)) > 3500))
  OR
  (c.rel1.MyClass2 IS NOT NULL and c.rel2.MyClass2 IS NOT NULL
  and ((c.myatt1 + SUM(c.rel1.MyClass2.myatt2)) > 12000));
  ```
- In SQL, in some cases set operations `UNION`, `INTERSECT` and `EXCEPT` in a block statement may fail to provide the exact result when the result is returned into one of the set variable
  ```
  set V1 = V1 UNION V2;
  ```
- In SQL, in some cases an attribute update in a block statement may fail to update some of the indexes defined on the sub classes.
- In the Enterprise Manager, when the Export Xml Dialog is open the Query Radio Button is not always correctly selected.

Resolved in Matisse 9.1.5

- The `TCPKEEPALIVE` parameter defines the keepalive control of the server `TCP/IP` connections.

- In the Enterprise Manager, the panels listing the schema class namespaces do not restore the previously selected namespace when returning to the panel.
- A `SQL INSERT` statement returns a `MATISSE-E-SYNTAX_ERROR` error when using the full qualified class name in `RETURNING REF()`.
  ```
  INSERT INTO myapp.Location (name) VALUES ('PER') RETURNING REF(myapp.Location) INTO per;
  ```
- In SQL, running a transaction deleting over 100,000 objects through a SQL method may fail with a `MATISSE-E-PSMABORT` error.
- On Linux, a Java program running Java `1.7.0_51` or higher may crash when the garbage collector executes the `com.matisse.sql.MtStatement.finalize()` method.
- In some rare cases, when a very large number of connections are opened and closed in parallel at the same time, the number of tokens which controls the maximum number of concurrent connections, may become inaccurate.
- When exporting the database schema in ODL or SQL DDL to a file, the file header does not always include the hostname of the server hosting the database.
- The `mt_xml` utility does not control the number of parallel tasks requested when exporting or importing XML documents.
- In some cases, the `mt_xml export` command with the `--task` option may fail to generate all `XML` documents leading to an import failure later on with the `MTXML-E-INVALXMLDOCSEQ` error.
- When starting a 32-bit database server with a page cache larger than 2Gbytes, the returned error `CFG-F-INVCACHEMAXCAP` is not specific enough.
- The `mt_server autocollect -d` command in verbose mode does not return the correct message.
- The `mt_connection count` with the `--full` option provides detailed connection count.
- On Linux RHEL 6.5 or higher, the status option for the matisse daemon script may fail.
  ```
  # /etc/init.d/matisse status
  status: Unknown job: mt_portmon
  ```

Resolved in Matisse 9.1.4

- The Matisse products are now supported on both Windows 8.1 and Windows Server 2012 R2.
- The `mt_datafile info` command now provides information about the maximum capacity of the datafile as well as about the OID representation format.
- In the Java binding, the `MtDatabase.getSchemaVersion()` method indicates if the database schema has been modified.
  ```
  public final int MtDatabase.getSchemaVersion()
  ```
- In the Java binding, the `MtObject.getMtOidToHexString()` method returns the hexadecimal representation of an OID.
  ```
  public final java.lang.String MtObject.getMtOidToHexString()
  ```

- In some cases a `SQL SELECT` statement may fail to filter on all the predicates when the same attribute is accessed by at least 2 different relationships and that an index defined on this attribute is selected by the SQL optimizer.
- In SQL in some cases a `GROUP BY` statement may fail to execute returning a `MATISSE-E-SQLRUNTIMEERR` error.
- In SQL in some cases an aggregate function in a `GROUP BY` statement does not compute the correct aggregate value.
- In SQL selecting an object from its pseudo-property `MtOID` fails with the `MATISSE-E-INVALID_NUM_VALUE` error if the value is a negative integer (i.e. `-83885944`) representing a large unsigned integer (i.e. `4211081352`).
- The `mt_sdl` utility may fail to export a database schema with shared properties defined in a namespace and used in multiple classes with indexes.

Resolved in Matisse 9.1.3

- In the Enterprise Manager Host panel, `Disks Capacity` shows the total disks capacity on the server.
- In the Java binding, the `com.matisse.MtDatabase` class implements the `AutoCloseable` interface.
- In the Java binding, the iterator classes `com.matisse.MtObjectIterator`, `com.matisse.MtPropertyIterator`, and `com.matisse.MtVersionIterator` implement the `AutoCloseable` interface.
- In the Java binding, one method in the `com.matisse.MtObjectFactory` interface has been updated as follows:
  `public Class<?> getJavaClass(String mtClsName)`
- In the Enterprise Manager, `Free Disk Space` in both the Host and Network Activity panels does show the total free space for all disks attached to the server.
- In some cases, XML export running multiple tasks in parallel may fail to export some object links for some bi-directional relationships with maximum cardinality of 1 on both side.
- In the Java binding, the `MtDatabase.removeVersion()` and `MtDatabase.getVersionFromName()` methods may fail to execute properly.
- In the C++ binding some functions manipulating fully qualified class names may generate memory leaks.
- In some cases the Linux installer program may fail to start `TCP` portmon if the `IPv6` stack has been disabled on the server.

Resolved in Matisse 9.1.2

- The Enterprise Manager Schema tree node icons visually differentiate inherited properties from locally defined ones.

- In the Enterprise Manager Database Object Browser Panel, the Statistics table content can be exported into a CSV file via the `Export To File` item of table popup menu.
- The Enterprise Manager Monitor and Datafiles Management Panels may indicate in the Status column of the Datafiles table that a datafile is collecting versions when it is not.
- In the Enterprise Manager Database Property Dialog, the horizontal scroll-bar of the Datafiles table does not properly move the column header after a datafile was added.
- In the Enterprise Manager Datafiles Management Panel, adding a primary or mirror datafile may fail.
- On Windows if a machine shuts down while Matisse servers are online. When the machine reboots the Matisse Port-monitor Service re-records the database servers while the server processes no longer exist.
- In rare cases, a database in-memory with multiple in-memory datafiles may fail to restart after a shutdown indicating that the datafile list order is incorrect.

Resolved in Matisse 9.1.1

- The `mt_server create` with the `--fullinit` option creates a new database configuration file and enables the `DATFULLINIT` parameter which defines the datafile initialization behavior.
- The `mt_server create` with the `--serverlog` option creates a new database configuration file and sets the server log maximum version number to a specific value.
- In the Enterprise Manager Datafile Panel, when issuing a remove datafile command the dialog box does not always list the correct datafile names.
- A SQL method fails to compile and to provide a meaningful error message when a class attribute or relationship and a local variable have the same name.
- In some cases, the compilation of the SQL method fails to report a meaningful error message when variable used into a `SELECT ... INTO` statement is not declared as an object selection.
- In the Java binding, the generated code for `lookupIdxName(...)` methods defined to query indexes is not specific enough for `MT_NUMERIC` type attribute.
- On Windows, the `mt_file list` command does not truncate the datafile name as expected.
- The `mt_file list` command always shows mirrored datafiles in wide mode (`--wide`).
- In some cases, when deleting a mirrored datafile the datafile name reported in the server log file is incorrect.
- In some cases, deleting a datafile on an active database may fail.
- In some cases adding a mirrored datafile to an online database, then removing the primary datafile and stopping the database before any activity on the mirrored datafile occurs may prevent the database to properly restart.
- In some cases, adding a mirrored disk on an active database may block the active transactions.

Resolved in Matisse 9.1.0

- Matisse DBMS provides full distributed XML-based asynchronous replication to complement the existing synchronous transactional replication capability.
- Matisse new database activity monitoring feature helps database administrators monitor the performance and availability of databases on a distributed network.
- Direct-io datafiles provide the performance and reliability that we normally see only in raw devices, but also hides the complexity of managing raw devices.
- A set of new optional database configuration parameters have been added to fine tune the performance of production databases.
- Servers, services, tools and utilities transparently support `IPv6`.
- In the Enterprise Manager, the Create and Update Database dialogs manage the `DATIODIRECT`, `AUTOCOLLECTFREQ`, `OBJTABLESIZ`, `OBJTABCLRFREQ` and `OBJTABCLRLEVEL` configuration parameters.
- The Enterprise Manager Monitoring Panel includes the Activity and History tab which show performance graphs providing insight into the performance over a period of time.
- The Enterprise Manager Import/Export XML task dialog provides progress of the running XML task.
- The Enterprise Manager log file records the login of the connected user.
- Matisse tools and utilities (`ODL`, `SQL`, `XML` and `DTS`) have been updated to be consistent with the other commands which support short and long name options as well as database source name compatible with `IPv6` address.
- The utility commands trace more execution information as the verbosity level increases.
- The `mt_xml export` command with the `--iobuffer` option writes XML documents in buffered I/O mode while the `--iostream` option writes XML documents in stream I/O mode. On Windows for example the buffered I/O mode is significantly faster than the stream I/O mode.
- The `mt_server logactivity` command allows you to manage the recordings of the database activity metrics.
- The `mt_server setrunlevel` command allows you to set the run level of automatic operations.
- The `mt_server setrunfrequency` command allows you to set the run frequency of automatic operations.
- The `mt_server monitor` command now provides information about the server cache hit rate.
- The `mt_file setextendsiz` command was renamed `setextendsize`.
- The `mt_server checklicense` with the `--info` option provides more information about the machine configuration to help issue customer license keys.
- On Windows, tools and utilities include description and version as well as company information.

- In some cases, when the Enterprise manager is unable to ping a database, the database status is set to `none` instead of `unreachable`.
- The execution of a `SELECT` statement fails with the `MATISSE-E-SQLRUNTIMEERR` error when `WHERE` clause include a `LIKE` expression followed by a relationship `COUNT` expression in this particular order.
  ```
  SELECT * FROM myclass WHERE myname LIKE '123' AND COUNT(mychildren) > 1
  ```
- On Windows, when an application opens a large number of connections at the same time one connection may fail with one of the following errors:
  ```
  MATISSE-F-NOPMADDR, Unable to get Port Monitor address
  MATISSE-F-SRVCONFAILED, Connection to server failed
  MATISSE-F-CONNTIMEOUT, Database "example" on host "localhost"
  ```
- On Windows in some cases after an upgrade of the Java JRE, the Matisse Enterprise Manager may fail to load the new Java Virtual Machine DLL leading to the need for reinstalling the Matisse product.

5 Documentation

Matisse documents available on the Web

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

- Installation guides for Linux, MacOS, 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 Objective-C Programmer's Guide* (and example applications)
- *Matisse C++ Programmer's Guide* (and example applications)
- *Matisse C API Reference*
- *Matisse ODL Programmer's Guide* (includes user's guide for `mt_sdl`)
- *Matisse Modeler 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`)

Documents included with Matisse standard installation

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

Open source bindings

- *Matisse Eiffel Programmer's Guide* (and example applications)
- *Matisse PHP Programmer's Guide* (and example applications)
- *Matisse Python Programmer's Guide* (and example applications)