

**A User's Guide to TDM:
The CRL TIPSTER Document Manager**

Nigel Sharples and Philip Bernick

MCCS-96-298

Computing Research Laboratory
Box 30001/3CRL
New Mexico State University
Las Cruces, New Mexico 88003

June 1996

*The Computing Research Laboratory was established by the
New Mexico State Legislature,
under the Science and Technology Commercialization Commission
as part of the Rio Grande Research Corridor.*

*A User's Guide to TDM:
The CRL TIPSTER
Document Manager*

MCCS-26-298

Nigel Sharples and Philip Bernick



**The Computing Research Laboratory
New Mexico State University
Las Cruces, NM 88003**

Communication regarding this documentation or TDM should be directed to
Nigel Sharples: nigel@crl.nmsu.edu

Copyright 1996 by the Computing Research Laboratory
All rights reserved.

CRL's TIPSTER Document Manager

TIPSTER

TIPSTER is an ARPA-funded program of research and development in information retrieval and extraction. Phase II of this program, which began in the spring of 1994, aims to facilitate the rapid transition of advances of these technologies into Government agencies.

The TIPSTER Architecture

One part of this effort involves the creation of a standard architecture (standard interfaces) for systems doing information retrieval and extraction. This architecture is being developed by a group of contractors and Government personnel (the TIPSTER Architecture Working Group), chaired by Ralph Grishman of NYU.

The architecture is described by an architecture design document, which undergoes regular revision. The most recent version of the TIPSTER Phase II Architecture Design Document, version 1.52, was prepared on July 7, 1995. The C-language header file, `tipster.h`, which specifies the C interface for TIPSTER operations, is included in the latest Design Document.

TDM: The CRL TIPSTER Document Manager

TDM, the TIPSTER Document Manager from the Computing Research Laboratory (CRL) at New Mexico State University, implements part of the TIPSTER architecture and provides a way to store multilingual documents with their annotations and attributes. Other software can then perform information retrieval, extraction and other tasks on text collections and documents.

TDM: The Current Release

This release of TDM inherits code and implementation philosophy from the previously released prototype TDM document manager. However, it has been rewritten to significantly improve robustness and usability.



Contents

PREFACE	<i>CRL's TIPSTER Document Manager.....</i>	<i>i</i>
	TIPSTER	<i>i</i>
	The TIPSTER Architecture	<i>i</i>
	TDM: The CRL TIPSTER Document Manager	<i>i</i>
	TDM: The Current Release	<i>i</i>
CHAPTER 1	<i>TDM: The CRL TIPSTER Document Manager</i>	<i>1</i>
	Background.....	<i>1</i>
	Definitions	<i>2</i>
	A Top-Down Look.....	<i>2</i>
	TDM Structure.....	<i>5</i>
	<i>Collections</i>	<i>5</i>
	<i>Documents</i>	<i>5</i>
	<i>Attributes and Values</i>	<i>6</i>
	<i>Annotations</i>	<i>6</i>
	<i>Spans</i>	<i>7</i>

	<i>Sets</i>	7
	<i>Reference</i>	7
	Summary.....	8
CHAPTER 2	<i>Using TDM: A Tutorial</i>	9
	Assumptions	9
	An Example Program	9
	Understand the Program	12
	<i>The Header Files</i>	12
	<i>TDM Configuration</i>	12
	<i>Destroy Old Test Collections</i>	12
	<i>Create a Test Collection and Attribute Set</i>	13
	<i>Create Documents for the Collection</i>	13
	<i>Open a Document and Create an Annotation</i>	14
	<i>Create an Attribute, Place it on a Document</i>	15
	<i>Save the Annotations and Attributes to the Document</i>	15
	Compile the Program.....	15
	Run the Program.....	16
CHAPTER 3	<i>Exception Handling</i>	17
	The TDM Exception Handling Package.....	17
	<i>Requirements</i>	17
	<i>Usage</i>	17
	The Throw and Catch Error Handling Mechanism	18
	Throw	19
	Fatal	20
	Other Functions	20
	Limitations	21
	Exception Patterns	22
CHAPTER 4	<i>TDM Functions</i>	23
	TDM Functions and Memory	23
	Table of TDM Functions.....	24



CHAPTER 5	<i>Compliance with the TIPSTER Standard.....</i>	61
	Deviations.....	61
	Additions to the TIPSTER Standard	61



TDM: The CRL TIPSTER Document Manager

This chapter contains:

- Some Background on TDM
- TDM Definitions
- A Top-Down Look at TDM
- TDM Structure

Background

CRL's TIPSTER document manger has been developed as part of ARPA's TIPSTER text project, and has been designed support document management (storage, retrieval, and manipulation) of documents, document collections, annotations, and attributes.

The TIPSTER Text project has a goal to develop a standard architecture that enables users to integrate information and extraction systems from a variety of sources, and, beyond this will allow these systems to share information and results. For example, a TIPSTER compliant text segmenter would apply annotation to the text that indicate text segments or word boundaries, etc. A name-finder could then use this data to add more annotations to indicate where names could be found in the document, and could assign attribute values to these annotations that indicate other information about the name like whether it is an organization name or a proper name. Of course, this kind of distinction could also be made by using different annoation types. Annotations are typically categorized into various types.

The Computing Research Laboratory (CRL) is using TDM in several of their research projects, including Oleada, a multilingual toolkit for language learners and instructors, and TEMPLE, a project in human assisted machine translation.

Definitions

Annotation: Annotations are information, like part-of-speech, comments, or other linguistic information, that are attached to a span of text in a document. An annotation contains a set of spans.

Attribute: Attributes are information, like a date, language, or other information that are attached to a document, annotation, or collection.

Collection: Collections contain documents. Collections cannot contain other collections.

Document: A document refers to an object that consists of text, its attributes and annotations.

Span: A span is a region, or range, of the document text. Spans are measured by character position.

Reference: A reference is a reference to an object, not an instance of the object, i.e., it is a pointer to an object. It is a way of storing enough information about an object to get to the object.

A Top-Down Look

These examples show user interfaces that use, but are not part of, TDM. Figure 1 shows a document manager interface that displays collections and a list of documents contained the highlighted Autosaves collection.

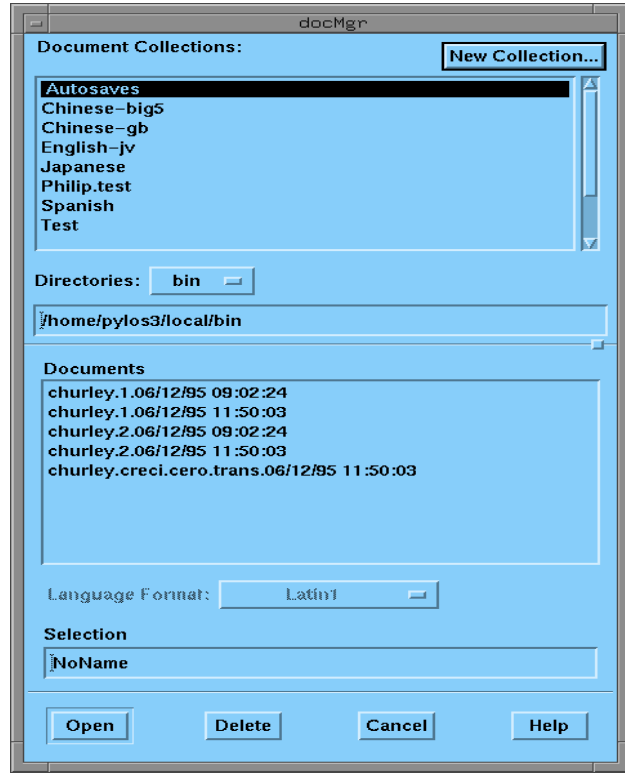


FIGURE 1. Document Manager User Interface Displaying Collections and Documents

Figure 2 shows a Spanish document that has some highlighted, or annotated text.

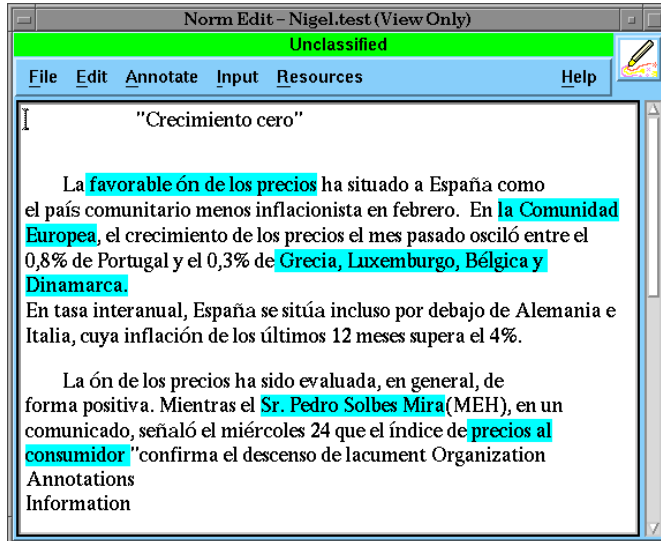


FIGURE 2. Spanish Text with Annotations

Each annotation shown in this document covers a single span. Annotations can also cover multiple spans. Figure 3 shows an interface that displays document attributes. Here the attributes are language mode, or language the text should be displayed in, security class, and permissions.

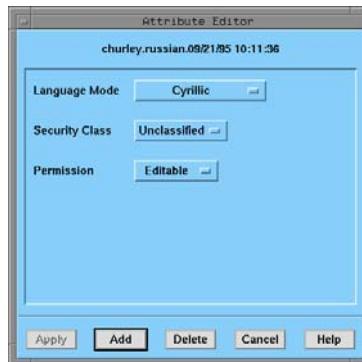


FIGURE 3. Interface Displaying Document Attributes

TDM Structure

Writing programs that use TDM is easier once TDM's structure is clear. The following describe TDM's structure in terms of objects. There two types of TDM objects, primitive, and non-primitive. Collections, documents, attributes, annotations, annotation and attribute values, spans, sets, and references are TDM objects.

Collections

The document manager can manage any number of collections. Collections cannot contain other collections. A collection consists of a name, the collection's attributes, and documents, as shown in Figure 4.

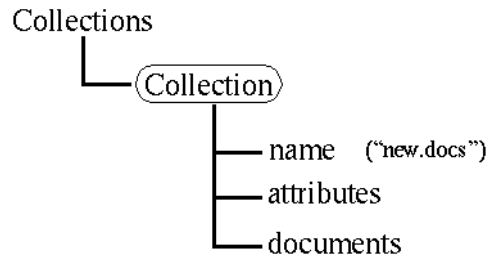


FIGURE 4. Collection Structure

Documents

Documents consist of a name, a document ID, a set or sequence of attributes, text, and a set or sequence of annotations, as shown in Figure 5.

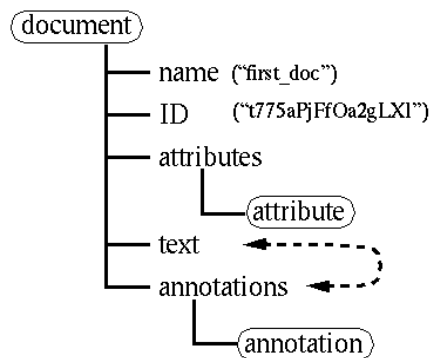


FIGURE 5. Document Structure

Attributes and Values

Attributes consist of an attribute name, and an attribute value. Values consist of a value type and data. Attribute and value structure is shown in Figure 6.

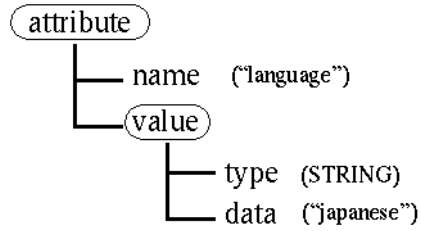


FIGURE 6. Attribute and Value Structure

Attribute values can be of several types. These types are: STRING, SEQUENCE (of attribute value), COLLECTION_REFERENCE, DOCUMENT_REFERENCE, ANNOTATION_REFERENCE, and ATTRIBUTE_REFERENCE

Annotations

Annotations consist of an ID, an annotation type, a sequence of spans, and a sequence of attributes, as shown in figure 7.

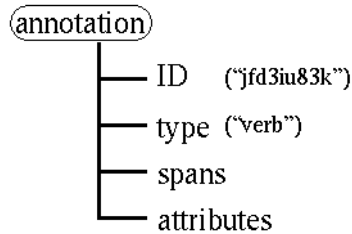


FIGURE 7. Annotation Structure

Spans

Spans consist of a beginning and ending byte position, as shown in Figure 8.

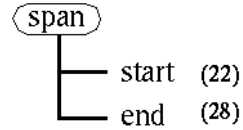


FIGURE 8. Span Structure

Sets

Sets are sequences of attributes, spans, annotations, strings, or attribute values, as shown in Figure 9.

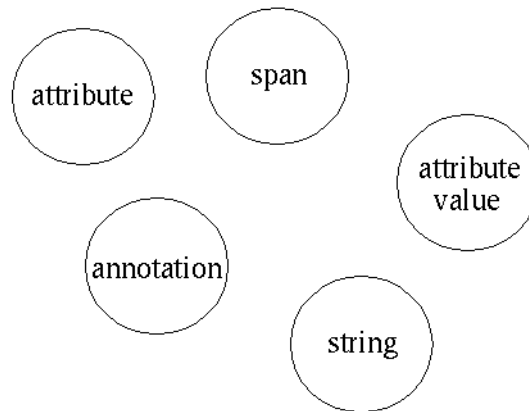


FIGURE 9. Set Structure

Reference

There are several kinds of reference, including collection references, document references, annotation references and attribute references. References are pointers to TDM objects, not instances of the object themselves. It is possible to have a refer-

ence to an object that no longer exists. Reference structures are shown in Figure 10.

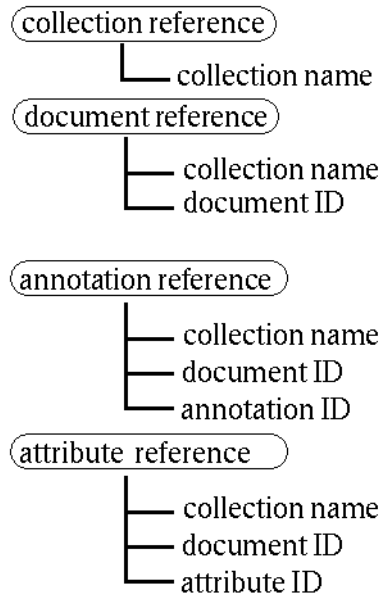


FIGURE 10. Reference Structures

Summary

This chapter provides only a cursory look at TDM, the TIPSTER document manager. More information on TDM is provided in the TDM Tutorial and in the TDM Function descriptions.

This chapter describes

- The Basic TDM Program
- An Example Program
- The Program Components
- Compiling and Executing the Program

Assumptions

This tutorial assumes that *tdm*, *gnu make*, *gcc*, *tcl*, *tcl-dp*, *tk*, *config*, *exception*, and *strlist* are installed, compiled, and available on the machine you're working on. If this is not the case, refer to the TDM installation document for obtaining these packages.

To use TDM you need to write a program. A basic program looks something like this:

```
config_Init("/home/app", "testapp", "1.0", "general");
tdm_Init();
```

config_Init initializes TDM 'configuration' variables, mainly to get the host and port (socket) to connect to.

An Example Program

You can experiment with TDM by using the following program.

```
#include <stdio.h>
#include <stdlib.h>
#include "rdm.h"
#include "config.h"
#define LIBRARY "TUTORIAL"
#include <exception.h>
```

```
int
main(int argc, char **argv)
{
    tdm_Collection col;
    tdm_Document doc;
    tdm_ByteSequence bcontents;
    tdm_Span span;
    tdm_SpanSet spans;
    tdm_Attribute att;
    tdm_AttributeSet atts;
    tdm_AttributeValue atval;
    tdm_Annotation ann;
    char *annid;
    char *config_file;
    /* get environment variable TDM_CONFIG, this should point
       to the tdm startup configuration file called tdm-startup-
       [version]-DOD where [version] is something like
       1.7.4. This file is below the TDM installation in a
       directory called config. */
    if((config_file = getenv("TDM_CONFIG")) == NULL)
    {
        fprintf(stderr, "Please set environment variable"
            "TDM_CONFIG to location of TDM configuration file\n");
        exit(1);
    }
    /* initialize config */
    config_Init_File(config_file);
    /* initialize TDM */
    tdm_Init();
    /* destroy testcol collection if it exists */
    try {
        tdm_Destroy("testcol");
    } catch("") {
    } endtry;
    col = tdm_CreateCollection("testcol", NULL);
    /* Create an attribute set with one attribute */
    atts = tdm_CreateAttributeSet();
    atval = tdm_CreateAttributeValue(STRING, "English");
    att = tdm_CreateAttribute("Language", atval);
    tdm_Push(att, atts);
    tdm_Free(atval);
    bcontents = tdm_CreateByteSequence("In the second division,"
        "the Keble eight, which could easily be mistaken for an"
        "OUBC boat, must surely win their blades and end up in"
        "the lower reaches of the first division.");
}
```

An Example Program

```
doc = tdm_CreateDocument(col, "doc1", bcontents, NULL,atts);
tdm_Free(doc);
tdm_Free(bcontents);
bcontents = tdm_CreateByteSequence("With Conservative MPs "
    "baying for action, he was well aware that the failure "
    "of veterinary experts to lift even partially the ban "
    "on beef on Monday night would be seen as yet another "
    "kick in the teeth.");
doc = tdm_CreateDocument(col, "doc2", bcontents, NULL,atts);
tdm_Free(doc);
tdm_Free(bcontents);
bcontents = tdm_CreateByteSequence("A QUARTER of all adults "
    "questioned in a survey of British history did not know "
    "the significance of 1066, two thirds had no idea who "
    "fought at Agincourt and more than half did not know who "
    "was involved in the Battle of Goose Green.");
doc = tdm_CreateDocument(col, "doc3", bcontents, NULL,atts);
tdm_Free(doc);
tdm_Free(bcontents);
tdm_Free(atts);
/* Open one of the documents we've just created */
doc = tdm_GetByExternalId(col, "doc2");
/* Create an annotation, first creating a span set and
attribute set. */
span = tdm_CreateSpan(5, 17);
spans = tdm_CreateSpanSet();
tdm_Push(spans, span);
atts = tdm_CreateAttributeSet();
ann = tdm_CreateAnnotation("party", spans,atts);
annid = tdm_AddAnnotation(doc, ann);
printf("ID of created annotation is %s\n", annid);
/* Create and put attribute on document */
atval = tdm_CreateAttributeValue(STRING, "22/5/96");
tdm_PutAttribute(doc, "date", atval);
/* free memory */
tdm_Free(atval);
tdm_Free(spans);
tdm_Free(atts);
tdm_Free(ann);
/* make sure annotations and attributes are saved to docu-
ment */
tdm_Commit(doc);
tdm_Free(doc);
tdm_Free(col);
return 0;
```

```
}
```

Understand the Program

There are several important pieces to this program.

The Header Files

```
#include <stdio.h>
#include <stdlib.h>
#include <rdm.h>
#include <config.h>
#define LIBRARY "TUTORIAL"
#include <exception.h>
```

The macro `LIBRARY` needs to be defined for the exception header file. Order is important: define the macro before including the exception header file.

TDM Configuration

The `TDM_CONFIG` environment variable should point to the tdm startup configuration file called `tdm-startup-[version]-DOD` where `[version]` is something like 1.7.4. When TDM was installed a directory called `config` should have been created that should contain this file.

```
    if((config_file = getenv("TDM_CONFIG")) == NULL)
    {
        fprintf(stderr, "Please set environment variable"
                "TDM_CONFIG to location of TDM configuration file\n");
        exit(1);
    }
    /* initialize config */
    config_Init_File(config_file);
    /* initialize TDM */
    tdm_Init();
```

Destroy Old Test Collections

We want to start with a clean slate. If this isn't the first time we've experimented with this test program, we need to destroy the old test collection. Otherwise, the program will try to recreate documents and a collection that already exist.

```
    /* destroy testcol collection if it exists */
    try {
        tdm_Destroy("testcol");
    } catch("") {
    } endtry;
```

Create a Test Collection and Attribute Set

This will create a test collection called `testcol`.

```
col = tdm_CreateCollection("testcol", NULL);
```

This will create an attribute set that contains the attribute `Language` whose attribute value is `English`.

```
atts = tdm_CreateAttributeSet();
atval = tdm_CreateAttributeValue(STRING, "English");
att = tdm_CreateAttribute("Language", atval);
tdm_Push(atts, att);
tdm_Free(atval);
```

Create Documents for the Collection

Here we will create three documents for the collection. In each case, `bcontents` is a `tdm` byte sequence consisting of the document text. `tdm_CreateDocument` needs five arguments: a collection object, a document name, document text, annotations, and attributes. `tdm_Free` is used to free the memory used to create the document.

```
bcontents = tdm_CreateByteSequence("In the second division, "
    "the Keble eight, which could easily be mistaken for an "
    "OUBC boat, must surely win their blades and end up in "
    "the lower reaches of the first division.");
doc = tdm_CreateDocument(col, "doc1", bcontents, NULL, atts);
tdm_Free(doc);
tdm_Free(bcontents);
bcontents = tdm_CreateByteSequence("With Conservative MPs "
    "baying for action, he was well aware that the failure "
    "of veterinary experts to lift even partially the ban "
    "on beef on Monday night would be seen as yet another "
    "kick in the teeth.");
doc = tdm_CreateDocument(col, "doc2", bcontents, NULL, atts);
tdm_Free(doc);
```

```
tdm_Free(bcontents);
bcontents = tdm_CreateByteSequence("A QUARTER of all adults"
    " questioned in a survey of British history did not know"
    " the significance of 1066, two thirds had no idea who "
    " fought at Agincourt and more than half did not know who"
    " was involved in the Battle of Goose Green.");
doc = tdm_CreateDocument(col, "doc3", bcontents, NULL,atts);
tdm_Free(doc);
```

We also need to free the memory used for `bcontents` and `atts`.

```
tdm_Free(bcontents);
tdm_Free(atts);
```

Open a Document and Create an Annotation

We'll use `tdm_GetByExternalId` to open one of the documents we've just created.

```
doc = tdm_GetByExternalId(col, "doc2");
```

Next we'll create an annotation. To do this, we first create span and attribute sets. The span will cover bytes 5 through 17. First we create a span, then we create a span set, then we push the span onto the span set.

```
span = tdm_CreateSpan(5, 17);
spans = tdm_CreateSpanSet();
tdm_Push(spans, span);
```

Next we create an attribute set.

```
atts = tdm_CreateAttributeSet();
```

Now we'll create an annotation, `party`. `tdm_CreateAnnotation` needs three arguments: annotation type, span set, and attribute set.

```
ann = tdm_CreateAnnotation("party", spans, atts);
```

The function `tdm_AddAnnotation` returns the annotation ID of the annotation that was added to the document. We'll also print out this annotation ID.

```
annid = tdm_AddAnnotation(doc, ann);
printf("ID of created annotation is %s\n", annid);
```

Create an Attribute, Place it on a Document

Next, we'll create an attribute and place it on a document. The function `tdm_CreateAttributeValue` gets two arguments: a value type, and attribute value data. The function `tdm_PutAttribute` has three arguments: a document, an attribute name, and an attribute value.

```
atval = tdm_CreateAttributeValue(STRING, "22/5/96");
tdm_PutAttribute(doc, "date", atval);
```

Now we'll free up the memory we've used to create these annotations and attributes.

```
tdm_Free(atval);
tdm_Free(spans);
tdm_Free(attribs);
tdm_Free(ann);
```

Save the Annotations and Attributes to the Document

Finally, we'll use `tdm_Commit` to save the annotations and attributes to the document. We'll also free up the memory we've used on the documents and the collection.

```
tdm_Commit(doc);
tdm_Free(doc);
tdm_Free(col);
return 0;
```

Compile the Program

If you've named the example program `example.c`, the following command should successfully compile it. `$` indicates the path prefix to the various includes, libraries, and other packages the program will use.

For Solaris

```
% gcc example.c -I$/tdm-1.7.4/include -L$/tdm-1.7.4/
lib -ltdm -lexception -lconfig -lstrlist -ldplite -
ltcl -lm -lsocket -lnsl -o example
```

For SunOS

```
% gcc example.c -I$/tdm-1.7.4/include -L$/tdm-1.7.4/
lib -ltdm -lexception -lconfig -lstrlist -ldplite -
ltcl -lm -o example
```

Once compiled you need to set the TDM_CONFIG variable, i.e.,

```
% setenv TDM_CONFIG $[tdm_installation_location]/
config/tdm-startup-[version]-DOD
```

Run the Program

Compiling the program will produce a file called `example`. Executing `example` should produce output that looks like this:

```
% example
Connected to TDM server version tdm-1.7.4-DOD on
hostname(port#) accessing:
TDM data: $/tdm[version]/data/Collections
client: $/tdm[version]/bin/rdm_client.tcl
ID of created annotation is A#1.05/23/6422
```

TDM data is a directory that holds the Collections directory. Under Collections is the documents directory that holds the documents created by executing this program.

This chapter describes

- The Basic TDM Error Handling Mechanism
- The TDM Exception Handling Package and its Requirements

The TDM Exception Handling Package

The TDM exception handling package allows users to define code to be executed if an exception occurs while executing protected code. The protected code and exception handling code is marked using the ‘try’ and ‘catch’ macros. Exceptions are raised with the ‘throw’ and ‘rethrow’ functions. An exception stack is maintained internally to monitor program execution through multiple exception handling blocks.

Requirements

The TDM Exception Handling Package requires gcc version 2.4 or greater to compile.

Usage

To use the Exception Handling Package, add ‘#include <exception.h>’ to the source file of the code needing to catch or throw exceptions. Be sure to define the cpp variable LIBRARY either in the *Makefile*, i.e.,

```
CFLAGS = -DLIBRARY=\"stdio\"
```

or before the `#include` statement.

The Throw and Catch Error Handling Mechanism

TDM uses a ‘throw’ and ‘catch’ error handling mechanism similar to C++. When TDM encounters an error, it calls ‘throw’ with an error type and message. The calling routine can intercept this with a ‘try’ ‘catch’ block.

The exception handling code has the following structure:

```
try {
    ... code that may cause an exception ...
} catch ("memory.bad_alloc"){
    ... handler for memory exceptions ...
} catch ("*") {
    ... handler for all other exceptions ...
} endtry
```

The try-endtry structure is called the ‘exception handling block’. A catch block or ‘handler’ specifies a pattern to match against raised exceptions, and code that executes when match is made. The ‘endtry’ at the end of the exception handling block is used to clean up some internal structures after the try, and to automatically rethrow an unmatched exception.

The try block holds normally executed code that may raise an exception. If no exception is raised, execution will branch to the end of the exception handling block (skipping all catch blocks). If an exception is raised, then the first catch block to match the raised exception is executed.

If another exception is not raised in the catch, execution will continue after the endtry. An exception that occurs in a catch will skip the remaining code and start looking for a matching catch in the exception handling block that appears just above the current catch block in the exception stack.

Note: An exception may be transferred to a different exception handling block if no catch statement in the current block holds a pattern that matches the exception. Using `catch ("*")` will ensure that at least one catch block will get any exception.

Throw

The first ‘catch’ in this example will only see errors of the type “memory.bad_alloc”, while the second will see all other errors. If the error isn’t caught the program will terminate.

Currently, most errors in TDM are generic; many of them are of type `documentServer.badArgument` or `documentServer.internalError`. What follows is a more complete description of TDM error handling.

Throw

```
void throw(const char *catch_ID, const char *format, ...);
```

An exception is raised using `throw()`. Parenthesis are necessary because `throw()` is a function. The first argument to `throw()` is a pattern that defines the exception being raised. The second argument is a printf style format string, and its optional arguments follow. The `throw()` function will force the program to jump into the first catch block of an established exception block which holds a pattern that matches the raised exception.

A new exception may be raised within a handler by calling `throw()`. If the programmer wants to reraise the exception the handler caught, then `rethrow()` should be called. The `rethrow()` function ought to be used only from within a catch block. Neither `throw()`, nor `rethrow()`, will ever return to the caller.

```
try{
    ... misc code that may cause an exception ...
} catch ("*") {
    if (we_really_want_this_exception){
        ... exception handling code ...
    } else rethrow();
} endtry
```

Fatal

A fatal exception is generated using the `fatal()` statement. As with `throw()`, a `printf` style format string and its arguments may be specified. It also may be used anywhere in the program. When `fatal` is called, the program will exit with an error message written to `stderr`. e.g., the `fatal()` call:

```
fatal("bad malloc(%d): %s", size, strerror(errno));
```

in the program may print something like:

```
<source file>:<line>: bad malloc(1234): out of memory
```

and exit (via `terminate()`). The `fatal()` function will never return to the caller, though installing a `terminate()` function to restart the program is fine as the exception stack is reset when `terminate()` is called.

Other Functions

The exception mechanism provides several functions that allow an exception block to gain access to information about the exception being handled:

```
const char *get_exception_message(void);  
const char *get_exception_code(void);  
int get_exception_errno(void);
```

The `'get_exception_message'` function returns a pointer to the message produced by the most recent `throw`. Accordingly the `'get_exception_code'` returns the full exception code thrown (see Exception Patterns below). The `'get_exception_errno'` returns the value of `errno` (from the C library) as it was set upon the most recently executed `throw()`.

A programmer can handle fatal errors by installing an alternate `'terminate'` function.

```
terminate_proc set_terminate(terminate_proc);
```

`'set_terminate'` is used to reset the `terminate` routine to a new function. A pointer to the old `terminate` routine is returned to the caller. The installed routine will be called whenever the function `terminate()` is called.

Limitations

The exception handling block is not a part of C, so some limitations exist. These limitations are:

1. Don't 'goto' a label outside the exception handling block from within the block. Using 'goto' to skip over the entire exception handling block is fine.
2. Don't 'goto' a label in an exception handling block from outside that block.
3. Don't ever use 'return' from within an exception handling block.
4. Don't use 'longjmp' to jump over an exception block. An example might help clarify this. If the call stack is A->B->C and B->C occurs within a try block in B, then C should NEVER use longjmp to go back to A. These operations will misalign the exception stack with the execution environment, possibly causing unexpected handlers to be called or exception stack underflow errors.

The exception mechanism will help you find code that may violate the above limitations. The 'set_exception_debug_file' function allows you to specify a file pointer to which debugging information about how the exception stack is working is written. The information written to this file includes:

1. Each entry to a try block is marked
2. Each exit from a try block is marked, along with whether a try block exited successfully, or if the block was exited after an exception was caught and handled.
3. Each throw and rethrow is marked, along with the exception and message being (re)thrown.
4. If an exception was raised, the location of each catch block that attempts to handle the exception is marked, along with information on whether the catch pattern matched the exception (causing it to be handled) or not.

The prototype for `set_exception_debug_file()` is:

```
FILE *set_exception_debug_file(FILE *);
```

The old debug file is returned, or NULL if no debug file was specified. NULL may also be passed as the argument to this function, in which case the debug messages will be disabled.

Exception Patterns

A raised exception has the general form:

```
<specifier1>.<specifier2>.<specifier3>.<...>
```

If a root of an exception pattern is given in a catch clause, that handler will accept any exception with that root. e.g., given the exception patterns “foo.bar.ack1” and “foo.bar.ack2”, catch (“foo.bar”) will handle both exceptions.

```
try {
    ... code that may cause an exception
    foo.bar.ack1 or foo.bar.ack2 or fee.bar...
} catch ("foo.bar"){
    ... This code is executed if either foo.bar.ack1
    or foo.bar.ack2 is raised...
} catch ("foo.bar.*"){
    Never gets executed because catch foo.bar above
    executes first. foo.bar is equivalent to
    foo.bar.* because there is an implicit * at the
    end.
} catch ("*.bar"){
    Only handles fee.bar because foo.bar handled the
    others...
} endtry
```

The astrisk (“*”) is a wildcard in that it will match everything up to a period (“.”). A partial specifier name may be given with a wildcard as well, though the use of this is limited. This behavior is similar to the way that file name matching is performed on most systems, with dots being the file separator (e.g., “/” under unix) between directory names.

catch (“foo.bar.*”) is equivalent to catch (“foo.bar”) in the example above. Also valid is catch (“*.bar”) which will match any exception with “bar” as its second specifier. Thus, a catch (“*”) block will handle all exceptions that reach it.

TDM Functions

Table of TDM Functions

tdm_AddAnnotation	tdm_AllAnnotations	tdm_AllAttributes
tdm_AnnotationReference	tdm_AnnotationsAt	tdm_AnnotationSet
tdm_AsString	tdm_AttributeReference	tdm_BreakDocLock
tdm_BreakWriteLock	tdm_Catalog	tdm_Closedown
tdm_CollectionCatalog	tdm_Commit	tdm_ConvertToString
tdm_CopyAttributes	tdm_CopyBareDocument	tdm_CopyDocument
tdm_CreateAnnotation	tdm_CreateAnnotationReference	tdm_CreateAnnotationSet
tdm_CreateAttribute	tdm_CreateAttributeReference	tdm_CreateAttributeSet
tdm_CreateAttributeValue	tdm_CreateAttributeValueSet	tdm_CreateByteSequence
tdm_CreateCollection	tdm_CreateCollectionReference	tdm_CreateDocument
tdm_CreateDocumentReference	tdm_CreateSpan	tdm_CreateSpanSet
tdm_CreateStringSet	tdm_CreateVolatileCollection	tdm_DeleteAnnotations
tdm_Destroy	tdm_Dup	tdm_FirstDocument
tdm_Free	tdm_GetAnnotation	tdm_GetAnnotationId
tdm_GetAnnotations	tdm_GetAttribute	tdm_GetAttributeName
tdm_GetAttributes	tdm_GetByExternalId	tdm_GetByExternalIdForRead
tdm_GetCollectionName	tdm_GetDocument	tdm_GetDocumentForRead
tdm_GetDocumentId	tdm_GetEnd	tdm_GetExternalId
tdm_GetId	tdm_GetName	tdm_GetOwner
tdm_GetParent	tdm_GetRawData	tdm_GetSpans
tdm_GetStart	tdm_GetType	tdm_GetValue
tdm_Init	tdm_IsDocLocked	tdm_IsLockClean
tdm_Length	tdm_LogStat	tdm_MergeAnnotations
tdm_NextAnnotations	tdm_NextDocument	tdm_OpenCollection
tdm_PutAnnotation	tdm_PutAttribute	tdm_Remove
tdm_RemoveAnnotation	tdm_RemoveAttribute	tdm_RemoveDocument
tdm_SelectAnnotations	tdm_ServerConnect	tdm_ServerDisconnect
tdm_ServerKill	tdm_SetAnnotations	tdm_SetAttributes
tdm_SetExternalId	tdm_SetId	tdm_SetOwner
tdm_SetSpans	tdm_SetString	tdm_ShowLockDoc
tdm_ShowLockDocAll	tdm_Sync	tdm_TypeOf

tdm_AddAnnotation

Add annotation to a document

Synopsis

```
tdm_String tdm_AddAnnotation(tdm_Document doc, tdm_Annotation ann);
```

Arguments

doc — A document

ann — Annotation

Returns

The annotation ID

Description

Add annotation to a document

Known Bugs

tdm_AllAnnotations

Return a list of all of the annotation ID's for a document.

Synopsis

```
char **tdm_AllAnnotations(tdm_Document doc);
```

Arguments

doc — A document

Returns

An array of strings, the annotation ID's. This can be freed with `free()`.

Description

Return a list of all of the annotation ID's for a document.

Known Bugs

Not tipster compliant.

tdm_AllAttributes

Get attribute names for object.

Synopsis

```
char **tdm_AllAttributes(void *obj);
```

Arguments

obj – A document, collection or annotation

Returns

An array of strings, the attribute names. This can be freed with `free()`.

Description

Returns a list of all attribute names which are associated with a particular document, collection or annotation.

Known Bugs

Not tipster compliant.

tdm_AnnotationsAt

Return annotations that start at byte position.

Synopsis

```
tdm_AnnotationSet tdm_AnnotationsAt(void *obj, int position);
```

Arguments

obj – A document or annotation set.

position – The byte position to start from within the document text.

Returns

An annotation set.

Description

Return annotations that start at position.

Known Bugs

tdm_AsString

Returns the literal contents of a document.

Synopsis

```
char *tdm_AsString(tdm_Document doc);
```

Arguments

doc – A document

Returns

The contents of a document.

Description

Returns the literal contents of a document.

Known Bugs

Not tipster compliant.

tdm_BreakDocLock

Break all locks on a document.

Synopsis

```
void tdm_BreakDocLock(tdm_Collection col, char *docname);
```

Arguments

col – A collection

docname – Document name

Returns

Description

Break all locks on a document.

Known Bugs

Not tipster compliant.

tdm_BreakWriteLock

Break write lock on a document and return the write locked document.

Synopsis

```
tdm_Document tdm_BreakWriteLock(tdm_Collection col, char *docname);
```

Arguments

col - Collection
docname - Document name

Returns

A document

Description

Break write lock on a document and return the write locked document.

Known Bugs

Not tipster compliant.

tdm_Catalog

Return a list of all document external id's.

Synopsis

```
char **tdm_Catalog(tdm_Collection col, char *Wildcard);
```

Arguments

col - Collection
wildcard - A wildcard such as test*

Returns

Array of strings, the document ID's.

Description

Return a list of all document external id's in a collection which match a glob style wildcard.

Known Bugs

Not tipster compliant.

tdm_Closedown

Destroy the TDM TCL interpreter.

Synopsis

```
void tdm_Closedown(void);
```

Arguments

Returns

Description

Destroy the TDM TCL interpreter.

Known Bugs

Not tipster compliant. We don't really know why anyone would want to use this.

tdm_CollectionCatalog

Return a list of all collections.

Synopsis

```
char **tdm_CollectionCatalog(char *Wildcard);
```

Arguments

Wildcard – wildcard pattern such as test*

Returns

Collection names as an array of strings.

Description

Return a list of all collections available to the server whose names match a glob style wildcard pattern.

Known Bugs

Not tipster compliant.

tdm_Commit

Commit the attributes and annotations of a document to the server.

Synopsis

```
void tdm_Commit(tdm_Document doc);
```

Arguments

doc – Document

Returns

Description

Commit the attributes and annotations of a document to the server. If any document attributes or annotations have been changed, this must be called to save them.

Known Bugs

Not tipster compliant.

tdm_ConvertToString

Convert a byte sequence to a string.

Synopsis

```
tdm_String tdm_ConvertToString(tdm_ByteSequence bseq);
```

Arguments

bseq – Byte sequence.

Returns

A string.

Description

Convert a byte sequence to a string.

Known Bugs

tdm_CopyAttributes

Copy attributes from one document to another.

Synopsis

```
void tdm_CopyAttributes(tdm_StringSet atts, tdm_Document from, tdm_Document to);
```

Arguments

atts – attributes

from – document to copy from.

to – document to copy to.

Returns

Description

Copy attributes from one document to another.

Known Bugs

Not tipster compliant.

tdm_CopyBareDocument

Copy a document and contents to the given collection.

Synopsis

```
tdm_Document tdm_CopyBareDocument(tdm_Collection new_parent, tdm_Document doc);
```

Arguments

new_parent – Collection to copy to.
doc – Document.

Returns

The newly created document.

Description

Copy a document to the given collection. Copy only document contents with no annotations or attributes.

Known Bugs

tdm_CopyDocument

Copy all of document to a new collection.

Synopsis

```
tdm_Document tdm_CopyDocument(tdm_Collection new_parent, tdm_Document doc);
```

Arguments

new_parent – The collection to copy document to.
doc – Document to copy.

Returns

The newly created document.

Description

Copy a document to a new collection, include annotations and attributes.

Known Bugs

tdm_CreateAnnotation

Create an annotation

Synopsis

```
tdm_Annotation tdm_CreateAnnotation(tdm_String ann_type, tdm_SpanSet spset,  
tdm_AttributeSet atset);
```

Arguments

ann_type – Annotation type.
spset – Span set.
atset – Attribute set.

Returns

Annotation. Users' responsibility to free.

Description

Create an annotation. Must supply the annotation type, a span set, and an attribute set.

Known Bugs

tdm_CreateAnnotationReference

Create an annotation reference.

Synopsis

```
tdm_AnnotationReference tdm_CreateAnnotationReference(tdm_Document doc,  
tdm_Annotation ann);
```

Arguments

doc – Document
ann – Annotation

Returns

Annotation reference.

Description

Create an annotation reference.

Known Bugs

tdm_CreateAnnotationSet

Create an annotation set.

Synopsis

```
tdm_AnnotationSet tdm_CreateAnnotationSet(void);
```

Arguments

Returns

Annotation set.

Description

Create an annotation set.

Known Bugs

tdm_CreateAttribute

Create an attribute.

Synopsis

```
tdm_Attribute tdm_CreateAttribute(char *name, tdm_AttributeValue atval);
```

Arguments

name – Name of attribute.

atval – Attribute value.

Returns

Attribute

Description

Create an attribute.

Known Bugs

tdm_CreateAttributeReference

Create an attribute reference.

Synopsis

```
tdm_AttributeReference tdm_CreateAttributeReference(tdm_Document doc, tdm_String  
attname);
```

Arguments

doc – Document.

attname – Attribute name.

Returns

Attribute reference.

Description

Create an attribute reference.

Known Bugs

tdm_CreateAttributeSet

Create an attribute set.

Synopsis

```
tdm_AttributeSet tdm_CreateAttributeSet(void);
```

Arguments

Returns

Attribute set.

Description

Create an attribute set.

Known Bugs

tdm_CreateAttributeValue

Create an attribute value.

Synopsis

```
tdm_AttributeValue tdm_CreateAttributeValue(enum tdm_AttributeValueType atype,  
void *val);
```

Arguments

atype – Attribute type.
val – value data.

Returns

Attribute value.

Description

Create an attribute value.

Known Bugs

tdm_CreateAttributeValueSet

Create an attribute value set.

Synopsis

```
tdm_AttributeValueSet tdm_CreateAttributeValueSet(void);
```

Arguments

Returns

Attribute value set.

Description

Create an attribute value set.

Known Bugs

tdm_CreateByteSequence

Create a byte sequence.

Synopsis

```
tdm_ByteSequence tdm_CreateByteSequence(tdm_String value);
```

Arguments

value – Contents of byte sequence.

Returns

Byte sequence.

Description

Create a byte sequence.

Known Bugs

tdm_CreateCollection

Create collection.

Synopsis

```
tdm_Collection tdm_CreateCollection(tdm_String name, tdm_AttributeSet attset);
```

Arguments

name – Collection name.

attset – Attribute set or NULL.

Returns

Collection.

Description

Create collection with optional attribute set.

Known Bugs

tdm_CreateCollectionReference

Create reference to a collection.

Synopsis

```
tdm_CollectionReference tdm_CreateCollectionReference(tdm_Collection col);
```

Arguments

col – Collection.

Returns

Collection reference.

Description

Create reference to a collection

Known Bugs

tdm_CreateDocument

Create a document.

Synopsis

```
tdm_Document tdm_CreateDocument(tdm_Collection col, tdm_String external_id,  
tdm_ByteSequence bseq, tdm_AnnotationSet annset, tdm_AttributeSet attset);
```

Arguments

col - Collection.
external_id - Document name.
bseq - Byte sequence (for document contents) or NULL.
annset - Annotation set or NULL.
attset - Attribute set or NULL.

Returns

Document.

Description

Create a document with optional contents, annotation set, attribute set.

Known Bugs

tdm_CreateDocumentReference

Create reference to a document.

Synopsis

```
tdm_DocumentReference tdm_CreateDocumentReference (tdm_Document doc);
```

Arguments

doc - Document.

Returns

Document reference.

Description

Create reference to a document.

Known Bugs

tdm_CreateSpan

Create a span.

Synopsis

```
tdm_Span tdm_CreateSpan(int start, int end);
```

Arguments

start – Start of span.

end – End of span.

Returns

Span

Description

Create a span.

Known Bugs

tdm_CreateSpanSet

Create a span set.

Synopsis

```
tdm_SpanSet tdm_CreateSpanSet(void);
```

Arguments

Returns

Span set.

Description

Create a span set.

Known Bugs

tdm_CreateStringSet

Create a string set.

Synopsis

```
tdm_StringSet tdm_CreateStringSet(void);
```

Arguments

Returns

String set.

Description

Create a string set.

Known Bugs

tdm_CreateVolatileCollection

Create a volatile collection.

Synopsis

```
tdm_Collection tdm_CreateVolatileCollection(tdm_AttributeSet attset);
```

Arguments

attset – Attribute set.

Returns

Collection.

Description

Create a volatile collection. This collection has no name.

Known Bugs

tdm_DeleteAnnotations

Delete annotations.

Synopsis

```
void tdm_DeleteAnnotations(void *obj, tdm_String ann_type, tdm_AttributeSet attset);
```

Arguments

obj – Document or annotation set.

ann_type – Annotation type or NULL.

attset – Attribute set or NULL.

Returns

Description

Delete annotations from a document or annotation set.

Known Bugs

tdm_Destroy

Destroy a persistent object.

Synopsis

```
void tdm_Destroy (tdm_String name);
```

Arguments

name – Collection name.

Returns

Description

Destroy a persistent object. Now just a collection.

Known Bugs

tdm_Dup

Copy an object.

Synopsis

```
void *tdm_Dup(void *obj);
```

Arguments

obj – A TDM object (collection, document, attribute, annotation, byte sequence, attribute value, span, any set).

Returns

Description

Copy an object.

Known Bugs

Not tipster compliant.

tdm_FirstDocument

Return first document from collection.

Synopsis

```
tdm_Document tdm_FirstDocument(tdm_Collection col);
```

Arguments

col – Collection

Returns

Document or NULL if no documents in collection.

Description

Return first document from collection. Return NULL if no documents.

Known Bugs

tdm_Free

Free an object.

Synopsis

```
void tdm_Free(void *obj);
```

Arguments

obj – An object

Returns

Description

Free an object.

Known Bugs

tdm_GetAnnotation

Get an annotation given a document or annotation set.

Synopsis

```
tdm_Annotation tdm_GetAnnotation(void *obj, char *aid);
```

Arguments

obj – Document or annotation set.
aid – Annotation ID.

Returns

Annotation or NULL if annotation doesn't exist.

Description

Get an annotation given a document or annotation set.

Known Bugs

tdm_GetAnnotationId

Get annotation ID from annotation reference.

Synopsis

```
tdm_String tdm_GetAnnotationId (tdm_AnnotationReference annref);
```

Arguments

annref – Annotation reference.

Returns

Annotation ID.

Description

Get annotation ID from annotation reference.

Known Bugs

tdm_GetAnnotations

Get all annotations from a document.

Synopsis

```
tdm_AnnotationSet tdm_GetAnnotations(tdm_Document doc);
```

Arguments

doc – Document.

Returns

Annotation set.

Description

Get all annotations from a document.

Known Bugs

tdm_GetAttribute

Returns the value of an attribute.

Synopsis

```
tdm_AttributeValue tdm_GetAttribute(void *obj, char *name);
```

Arguments

obj – Annotation, collection or document.
name – Attribute name.

Returns

Attribute value or NULL if attribute doesn't exist.

Description

Return the value of an attribute.

Known Bugs

tdm_GetAttributeName

Get attribute name from attribute reference.

Synopsis

```
tdm_String tdm_GetAttributeName(tdm_AttributeReference attref);
```

Arguments

attref – Attribute reference.

Returns

Attribute name.

Description

Get attribute name from attribute reference.

Known Bugs

tdm_GetAttributes

Get all attributes from a document, collection or annotation.

Synopsis

```
tdm_AttributeSet tdm_GetAttributes(void *obj);
```

Arguments

obj – Document, collection or annotation.

Returns

Attribute set.

Description

Get all attributes from a document, collection or annotation.

Known Bugs

tdm_GetByExternalId

Open a document for write access.

Synopsis

```
tdm_Document tdm_GetByExternalId(tdm_Collection col, char *docname);
```

Arguments

col – Collection.
docname – Document name.

Returns

Document.

Description

Open a document for write access. Raise exception on failure. If the document has already been read locked on the same socket, this will ‘upgrade’ the lock to a write lock. If the document has already been write locked on the same socket, return the document without changing the lock.

Known Bugs

tdm_GetByExternalIdForRead

Open a document for read access.

Synopsis

```
tdm_Document tdm_GetByExternalIdForRead(tdm_Collection col, char *docname);
```

Arguments

col – Collection.
docname – Document name.

Returns

Description

Open a document for read access. Raise exception on failure. If the document has already been read or write locked by the same socket, return the document without changing the lock.

Known Bugs

Not tipster compliant.

tdm_GetCollectionName

Get collection name from a reference object.

Synopsis

```
tdm_String tdm_GetCollectionName(void *obj);
```

Arguments

obj – Annotation, collection, document or attribute reference.

Returns

Collection name.

Description

Get collection name from a reference object.

Known Bugs

tdm_GetDocument

Get a document from its internal ID.

Synopsis

```
tdm_Document tdm_GetDocument(tdm_Collection col, tdm_String doc_id);
```

Arguments

col – Collection.

doc_id – Document ID.

Returns

Document.

Description

Get a document from its internal ID.

Known Bugs

tdm_GetDocumentForRead

Get a read locked document from its internal ID.

Synopsis

```
tdm_Document tdm_GetDocumentForRead(tdm_Collection col, tdm_String doc_id);
```

Arguments

col – Collection.
doc_id – Document ID.

Returns

Document.

Description

Get a read locked document from its internal ID.

Known Bugs

Not tipster compliant.

tdm_GetDocumentId

Get document internal ID.

Synopsis

```
tdm_String tdm_GetDocumentId(void *obj);
```

Arguments

obj – Document or annotation reference.

Returns

Document internal ID.

Description

Get document internal ID from a document reference or annotation reference.

Known Bugs

tdm_GetEnd

Get end of span.

Synopsis

```
int tdm_GetEnd(tdm_Span span);
```

Arguments

span – Span.

Returns

End of span.

Description

Get end of span.

Known Bugs

tdm_GetExternalId

Get external ID of a document.

Synopsis

```
char *tdm_GetExternalId(tdm_Document doc);
```

Arguments

doc – Document.

Returns

Document external ID.

Description

Get external ID of a document.

Known Bugs

tdm_GetId

Get internal ID of a document.

Synopsis

```
tdm_String tdm_GetId(void *obj);
```

Arguments

obj – Document or annotation.

Returns

Document internal ID.

Description

Get internal ID of a document or annotation.

Known Bugs

tdm_GetName

Get name of a collection or attribute.

Synopsis

```
tdm_String tdm_GetName(void *obj);
```

Arguments

obj – Collection or Attribute.

Returns

Object name.

Description

Get name of a collection or attribute.

Known Bugs

tdm_GetOwner

Get owner of collection.

Synopsis

```
tdm_String tdm_GetOwner(tdm_Collection col);
```

Arguments

col – Collection.

Returns

Owner.

Description

Get owner of collection.

Known Bugs

tdm_GetParent

Get parent collection of a document.

Synopsis

```
tdm_Collection tdm_GetParent(tdm_Document doc);
```

Arguments

doc – Document.

Returns

Collection.

Description

Get parent collection of a document.

Known Bugs

tdm_GetRawData

Get document contents.

Synopsis

```
tdm_ByteSequence tdm_GetRawData(tdm_Document doc);
```

Arguments

doc – Document.

Returns

Document contents as a byte sequence.

Description

Get document contents as a byte sequence.

Known Bugs

tdm_GetSpans

Get spans from annotation.

Synopsis

```
tdm_SpanSet tdm_GetSpans(tdm_Annotation ann);
```

Arguments

ann — Annotation.

Returns

Spans

Description

Get spans from annotation

Known Bugs

tdm_GetStart

Get start of span.

Synopsis

```
int tdm_GetStart(tdm_Span span);
```

Arguments

span — Span.

Returns

Start of span.

Description

Get start of span.

Known Bugs

tdm_GetType

Get annotation type

Synopsis

```
tdm_String tdm_GetType(tdm_Annotation ann);
```

Arguments

ann — Annotation.

Returns

Annotation type.

Description

Get annotation type

Known Bugs

tdm_GetValue

Get attribute or attribute value.

Synopsis

```
void *tdm_GetValue(void *obj);
```

Arguments

obj – Attribute value or attribute.

Returns

Return a `tdm_AttributeValue` if argument was a `tdm_Attribute`, otherwise return a `tdm_AttributeReference`, `tdm_AnnotationReference`, `tdm_AttributeValueSet`, `tdm_String`, `tdm_CollectionReference`, or `tdm_DocumentReference`.

Description

Get value of attribute value, or attribute.

Known Bugs

tdm_Init

Initialize TDM.

Synopsis

```
void tdm_Init(void);
```

Arguments

Returns

Description

Initialize TDM. Must be called before any other TDM function calls. The configuration values `tdm_host` and `tdm_port` must be set before this function is called.

Known Bugs

Not tipster compliant.

tdm_IsDocLocked

Determine whether a document is locked.

Synopsis

```
int tdm_IsDocLocked(tdm_Collection col, char *docname);
```

Arguments

col – Collection.
docname – Document name.

Returns

1 if document locked, 0 otherwise.

Description

Determine whether a document is locked.

Known Bugs

Not tipster compliant.

tdm_IsLockClean

Determine whether a lock is clean.

Synopsis

```
int tdm_IsLockClean(void *obj);
```

Arguments

obj – Document

Returns

1 if lock is clean, 0 otherwise.

Description

Determine whether a lock is clean. A lock is clean when there is no write lock on the document.

Known Bugs

Not tipster compliant.

tdm_Length

Return length of a set or collection.

Synopsis

```
int tdm_Length(void *obj);
```

Arguments

obj – Set or Collection.

Returns

Length.

Description

Return length of a set or collection.

Known Bugs

tdm_LogStat

Switch server logging on or off.

Synopsis

```
void tdm_LogStat(int setting);
```

Arguments

setting – Zero is off, anything else is on.

Returns

Description

Switch server logging on or off.

Known Bugs

Not tipster compliant.

tdm_MergeAnnotations

Merge annotation sets.

Synopsis

```
tdm_AnnotationSet tdm_MergeAnnotations(tdm_AnnotationSet annset1,  
tdm_AnnotationSet annset2);
```

Arguments

annset1 – First annotation set.

annset2 – Second annotation set.

Returns

New annotation set.

Description

Merge two annotation sets. Assume that the annotation ID's change.

Known Bugs

tdm_NextAnnotations

Return annotations that start at position or greater than position.

Synopsis

```
tdm_AnnotationSet tdm_NextAnnotations(void *obj, int position);
```

Arguments

obj – Document or annotation set.

position – Character position within document text.

Returns

Annotation set.

Description

Return annotations that start at position or greater than position.

Known Bugs

tdm_NextDocument

Return next document from collection.

Synopsis

```
tdm_Document tdm_NextDocument(tdm_Collection col);
```

Arguments

col – Collection.

Returns

Document, or NULL if no more documents.

Description

Return next document from collection.

Known Bugs

tdm_OpenCollection

Open a collection.

Synopsis

```
tdm_Collection tdm_OpenCollection(char *name);
```

Arguments

name – Collection name.

Returns

Collection.

Description

Open a collection.

Known Bugs

tdm_PutAnnotation

Attaches a new annotation to a document.

Synopsis

```
char *tdm_PutAnnotation(tdm_Document doc, tdm_SpanSet spans, tdm_AttributeSet attributes, tdm_String type);
```

Arguments

doc - Document.
spans - Span set.
attributes - Attribute set.
type - Annotation type.

Returns

Annotation ID of created annotation.

Description

Attaches a new annotation to a document.

Known Bugs

Not tipster compliant.

tdm_PutAttribute

Attaches an attribute to an object.

Synopsis

```
void tdm_PutAttribute(void *obj, char *attname, tdm_AttributeValue atval);
```

Arguments

obj - Collection, document or annotation.
attname - Attribute name.
atval - Attribute value.

Returns

Description

Attaches an attribute to a collection, document or annotation.

Known Bugs

Not tipster compliant.

tdm_Remove

Remove an object from a set.

Synopsis

```
void tdm_Remove(void *set, int index);
```

Arguments

set – Set.
index – Index into set from where to remove object.

Returns

Description

Remove an object from a set.

Known Bugs

Not tipster compliant.

tdm_RemoveAnnotation

Remove an annotation.

Synopsis

```
void tdm_RemoveAnnotation(void *obj, tdm_String aid);
```

Arguments

obj – Document or annotation set.
aid – Annotation ID.

Returns

Description

Remove an annotation from a document or annotation set.

Known Bugs

tdm_RemoveAttribute

Remove an attribute from an object.

Synopsis

```
void tdm_RemoveAttribute(void *obj, char *attname);
```

Arguments

obj – Annotation, collection, or document.
attname – Attribute name.

Returns

Description

Remove an attribute from an object.

Known Bugs

Not tipster compliant.

tdm_RemoveDocument

Remove document given document ID.

Synopsis

```
void tdm_RemoveDocument(tdm_Collection col, tdm_String doc_id);
```

Arguments

col – Collection.
doc_id – Document ID.

Returns

Description

Remove document given document ID.

Known Bugs

tdm_SelectAnnotations

Select annotations.

Synopsis

```
tdm_AnnotationSet tdm_SelectAnnotations(void *obj, tdm_String ann_type,  
tdm_AttributeSet specs);
```

Arguments

obj – Document or annotation set.
ann_type – Annotation type or NULL. If NULL then don't limit what type of annotation will be selected.
specs – Attribute set of specifications or NULL. For annotations to be selected their attributes must exactly match these specifications.

Returns

Annotation set.

Description

Select annotations from a document or annotation set.

Known Bugs

tdm_ServerConnect

Connect to TDM server.

Synopsis

void tdm_ServerConnect(void);

Arguments

Returns

Description

Connect to TDM server, using configuration values. You must set the server host and port by setting the configuration variables tdm_host and tdm_port.

Known Bugs

Not tipster compliant.

tdm_ServerDisconnect

Disconnect from current TDM server.

Synopsis

void tdm_ServerDisconnect(void);

Arguments

Returns

Description

Disconnect from current TDM server.

Known Bugs

Not tipster compliant.

tdm_ServerKill

Kill the TDM server that you are currently connected to.

Synopsis

void tdm_ServerKill(void);

Arguments

Returns

Description

Kill the TDM server that you are currently connected to.

Known Bugs

Not tipster compliant.

tdm_SetAnnotations

Put annotation set on document.

Synopsis

```
void tdm_SetAnnotations(tdm_Document doc, tdm_AnnotationSet annset);
```

Arguments

doc – Document.
annset – Annotation set.

Returns

Description

Put annotation set on document. This will erase any annotations that are currently on the document.

Known Bugs

Not tipster compliant.

tdm_SetAttributes

Put an attribute set on an object.

Synopsis

```
void tdm_SetAttributes(void *obj, tdm_AttributeSet attset);
```

Arguments

obj – Collection, document or annotation.
attset – Attribute set.

Returns

Description

Put an attribute set on an object.

Known Bugs

Not tipster compliant.

tdm_SetExternalId

Set external ID of a document.

Synopsis

```
void tdm_SetExternalId(tdm_Document doc, tdm_String id);
```

Arguments

doc – Document.
id – Document name.

Returns

Description

Set external ID of a document.

Known Bugs

tdm_SetId

Set annotation ID.

Synopsis

```
void tdm_SetId(tdm_Annotation ann, char *id);
```

Arguments

ann – Annotation.
id – Annotation ID.

Returns

Description

Set annotation ID

Known Bugs

Not tipster compliant.

tdm_SetOwner

Set owner of a collection.

Synopsis

```
void tdm_SetOwner(tdm_Collection col, tdm_String owner);
```

Arguments

col – Collection.
owner – Collection owner.

Returns

Description

Set owner of a collection.

Known Bugs

tdm_SetSpans

Set spans on a annotation

Synopsis

```
void tdm_SetSpans(tdm_Annotation ann, tdm_SpanSet spset);
```

Arguments

ann – Annotation.
spset – Span set.

Returns

Description

Set spans on a annotation

Known Bugs

Not tipster compliant.

tdm_SetString

Set contents of document.

Synopsis

```
void tdm_SetString(tdm_Document doc, char *contents);
```

Arguments

doc – Document.
contents – Document contents.

Returns

Description

Updates the document to contain the specified contents.

Known Bugs

Not tipster compliant.

tdm_ShowLockDoc

Show all locks on the document.

Synopsis

```
char **tdm_ShowLockDoc(tdm_Collection col, char *docname, int *count);
```

Arguments

col – Collection.
docname – Document name.
count – Number of items in returned array.

Returns

Lock information for this document.

Description

Show all locks on the document.

Known Bugs

Not tipster compliant.

tdm_ShowLockDocAll

Show all documents locked by the server.

Synopsis

```
char **tdm_ShowLockDocAll(int *count);
```

Arguments

count – Number if items in returned array.

Returns

Lock information for all locked documents.

Description

Show all documents locked by the server.

Known Bugs

Not tipster compliant.

tdm_Sync

Synchronize a persistent object.

Synopsis

```
void tdm_Sync(void *obj);
```

Arguments

obj – Persistent object (collection).

Returns

Description

Synchronize a persistent object, now just a collection.

Known Bugs

tdm_TypeOf

Return the type of an attribute value.

Synopsis

```
enum tdm_AttributeValueType tdm_TypeOf(tdm_AttributeValue attval);
```

Arguments

attval – Attribute value.

Returns

Type of attribute value.

Description

Return the type of an attribute value.

Known Bugs

Compliance with the TIPSTER Standard

This chapter describes

- How TDM Differs From the TIPSTER Architecture Standard
- Additions to the TIPSTER Standard in TDM

Deviations

You must call `tdm_Init()` before any other TDM function calls.

To save annotations that you have changed, or document attributes, call `tdm_Commit()`.

Additions to the TIPSTER Standard

`tdm_AllAttributes` returns an array of strings of attribute names.

`tdm_Dup` duplicates a TDM object

`tdm_GetDocumentForRead`, `tdm_GetByExternalIdForRead`. TDM uses a document locking mechanism You can open a document for read or write. The standard functions `tdm_GetDocument` and `tdm_GetDocumentByExternalId` return write-locked documents. The non-standard calls `tdm_GetDocumentForRead`, `tdm_GetByExternalIdForRead` return a read-locked document.

There can only one write lock on a document, but many read locks. If a document is opened (for read or write), you will not be able to open it again with a write-lock; you will succeed in opening it with a read-lock. The lock is freed with a call to `tdm_Free`.

`tdm_BreakDocLock` breaks all locks on a document.

`tdm_ShowLockDoc` returns an array of strings that are the locks on a document.

`tdm_ShowLockDocAll` returns an array of strings that are all locked documents in the TDM server.

`tdm_IsDocLocked` tells you whether a document is locked or not.

`tdm_BreakWriteLock` breaks the current write lock on a document and returns a write locked document.

`tdm_IsLockClean` tells you whether the lock on a document is clean. A clean lock means there is no write lock on the document and you are assured the document contents/annotations/attributes won't change.

`tdm_Catalog` returns an array of strings of external ID's in a collection. You can't have document external ID's that are the same within a collection.

`tdm_SetAttributes` set the attributes for an object.

`tdm_SetAnnotations` puts an annotation set on a document.

`tdm_AsString` returns the contents of a document as a string.

`tdm_SetString` sets the contents of a document.

`tdm_CopyAttributes` copies attributes from one document to another.

`tdm_AllAnnotations` returns an array of strings of annotation ID's for a document.

`tdm_PutAnnotation` puts an annotation on a document.

`tdm_SetAttributes` puts an attribute set on an object

`tdm_Remove` removes an element from a set.

`tdm_CollectionCatalog` returns an array of strings of collection names.

`tdm_CloseDown` disconnects from the TDM server and destroys the TCL interpreter.

`tdm_LogStat` switches logging on or off.

`tdm_ServerConnect` connects to a TDM server.

`tdm_ServerDisconnect` disconnects from a TDM server.

`tdm_ServerKill` kills the TDM server.

Index

A

Annotation 2, 6, 25
annotation start position 51
ANNOTATION_REFERENCE 6
Attach a new annotation to a
 document 52
Attach an attribute to an object 52
Attribute 2
attribute names 26
Attribute value types 6
ATTRIBUTE_REFERENCE 6
Attributes and Values 6

B

break locks 27
break write lock 28
byte position 26

C

catch 18, 22
Catch block 18
Collection length 50

COLLECTION_REFERENCE 6
Collections 2, 5
Commit attributes and annotations 29
Compiling the Example Program 15
config.h 12
config_Init 9
configuration variables 9
Connect to TDM server 55
Convert a byte sequence 30
Copy a document and contents to the
 given collection 31
copy all of document 31
Copy an object 39
Copy attributes 30
copy document and contents 31
Create a byte sequence 35
Create a Collection 13
Create a Document 36
Create a span 37
Create a span set 37
Create a string set 37
Create a volatile collection 38

- Create an annotation 32
- Create an annotation reference 32
- Create an annotation set 33
- Create an attribute 33
- Create an attribute reference 33
- Create an attribute set 34
- Create an attribute value 34
- Create an attribute value set 34
- Create collection 35
- Create reference to a collection 35
- Create reference to a document 36

- D**
- Delete annotations 38
- Destroy a persistent object 38
- Destroy the TDM TCL interpreter 29
- Determine whether a lock is clean 49
- Disconnect from current TDM server 55
- Document 2
- document contents 27
- Document locking 49
- DOCUMENT_REFERENCE 6
- Documents 5
- documentServer.badArgument 19
- documentServer.internalError 19

- E**
- endtry 18, 22
- Error Handling 17
- Exception Handling 17
- exception handling block 18
- Exception Handling Package 17
- Exception Patterns 22
- exception.h 12
- external id's 28

- F**
- Fatal Exceptions 20
- first document from collection 39
- free 25, 26
- Free an object 39

- G**
- Get a document from its' internal ID 43
- Get a read locked document from its' internal ID 44
- Get all annotations from a document 40
- Get all attributes from a document 41
- Get an annotation given a document or annotation set 40
- Get annotation ID 40
- Get annotation type 47
- Get attribute name from attribute reference 41
- Get attribute or attribute value 48
- Get collection name from a reference object 43
- Get document contents 46
- Get document internal ID 44, 45
- Get end of span 44
- Get name of a collection or attribute 45
- Get owner of collection 46
- Get parent collection of a document 46
- Get spans from annotation 47
- Get start of span 47

- H**
- Header Files 12

- I**
- Initialize TDM 48

- K**
- Kill the TDM server 55

- L**
- LIBRARY macro 12
- list collections 29

- M**
- Memory 23
- Merge annotation sets 50

- O**
- Open a collection 51
- Open a document for read 42
- Open a document for write 42

Other Exception Functions 20

P

Put an attribute set on an object 56

Put annotation set on document 56

R

rdm.h 12

Reference 2, 7

Remove an annotation 53

Remove an attribute from an object 53

Remove an object from a set 53

Remove document given document
ID 54

Return next document from
collection 51

Return the value of an attribute 41

S

Select annotations 54

SEQUENCE 6

server logging 50

Set annotation ID 57

Set external ID of a document 57

Set owner of a collection 57

Sets 7

Show all documents locked by the
server 59

Span 2

Spans 7

stdio.h 12

stdlib.h 12

STRING 6

T

TCL interpreter 29

tdm 33, 36

TDM Basic Program 9

TDM Functions 23

TDM non-primitive types 23

TDM objects 39

TDM primitive type 23

tdm_AddAnnotation 14, 25

tdm_AllAnnotations 25

tdm_AllAttributes 26

tdm_Annotation 23, 25, 32, 40, 47, 57,
58

tdm_AnnotationReference 32, 40, 48

tdm_AnnotationsAt 26

tdm_AnnotationSet 26, 33, 36, 40, 50,
51, 54, 56

tdm_AsString 27

tdm_Attribute 33, 48

tdm_AttributeReference 33, 41

tdm_AttributeSet 32, 34, 35, 36, 38, 41,
52, 54, 56

tdm_AttributeValue 23, 33, 34, 41, 48,
52, 60

tdm_AttributeValueSet 34, 48

tdm_AttributeValueType 34, 60

tdm_BreakDocLock 27

tdm_BreakWriteLock 28

tdm_ByteSequence 30, 35, 36

tdm_Catalog 28

tdm_Closedown 29

tdm_Collection 23, 27, 28, 31, 35, 36,
38, 39, 42, 43, 44, 46, 49, 51, 54,
57, 59

tdm_CollectionCatalog 29

tdm_CollectionReference 35, 48

tdm_Commit 15, 29

TDM_CONFIG 12, 16

tdm_ConvertToString 30

tdm_CopyAttributes 30

tdm_CopyBareDocument 31

tdm_CopyDocument 31

tdm_CreateAnnotation 14, 32

tdm_CreateAnnotationReference 32

tdm_CreateAnnotationSet 33

tdm_CreateAttribute 13, 33

tdm_CreateAttributeReference 33

tdm_CreateAttributeSet 13, 14, 34

tdm_CreateAttributeValue 13, 15, 34

tdm_CreateAttributeValueSet 34

tdm_CreateByteSequence 13, 14, 35

tdm_CreateCollection 13, 35

tdm_CreateCollectionReference 35

tdm_CreateDocument 13, 14, 36

tdm_CreateDocumentReference 36

tdm_CreateSpan 14, 37

tdm_CreateSpanSet 14, 37
tdm_CreateStringSet 37
tdm_CreateVolatileCollection 38
tdm_DeleteAnnotations 38
tdm_Destroy 13, 38
tdm_Document 23, 25, 27, 28, 29, 30,
31, 32, 33, 36, 39, 40, 42, 43, 44,
45, 46, 51, 52, 56, 57, 58
tdm_DocumentReference 36, 48
tdm_Dup 39
tdm_FirstDocument 39
tdm_Free 13, 14, 39
tdm_GetAnnotation 40
tdm_GetAnnotationId 40
tdm_GetAnnotations 40
tdm_GetAttribute 41
tdm_GetAttributeName 41
tdm_GetAttributes 41
tdm_GetByExternalId 14, 42
tdm_GetByExternalIdForRead 42
tdm_GetCollectionName 43
tdm_GetDocument 43
tdm_GetDocumentForRead 44
tdm_GetDocumentId 44
tdm_GetEnd 44
tdm_GetExternalId 45
tdm_GetId 45
tdm_GetName 45
tdm_GetOwner 46
tdm_GetParent 46
tdm_GetRawData 46
tdm_GetSpans 47
tdm_GetStart 47
tdm_GetType 47
tdm_GetValue 48
tdm_Init 12, 48
tdm_IsDocLocked 49
tdm_IsLockClean 49
tdm_Length 50
tdm_LogStat 50
tdm_MergeAnnotations 50
tdm_NextAnnotations 51
tdm_NextDocument 51
tdm_OpenCollection 51
tdm_Push 13, 14
tdm_PutAnnotation 52
tdm_PutAttribute 15, 52
tdm_Remove 53
tdm_RemoveAnnotation 53
tdm_RemoveAttribute 53
tdm_RemoveDocument 54
tdm_SelectAnnotations 54
tdm_ServerConnect 55
tdm_ServerDisconnect 55
tdm_ServerKill 55
tdm_SetAnnotations 56
tdm_SetAttributes 56
tdm_SetExternalId 57
tdm_SetId 57
tdm_SetOwner 57
tdm_SetSpans 58
tdm_SetString 58
tdm_ShowLockDoc 59
tdm_ShowLockDocAll 59
tdm_Span 23, 37, 44, 47
tdm_SpanSet 32, 37, 47, 52, 58
tdm_String 25, 30, 32, 33, 35, 36, 38, 40,
41, 43, 44, 45, 46, 47, 48, 52, 53,
54, 57
tdm_StringSet 30, 37
tdm_Sync 60
tdm_TypeOf 60
Throw 18, 19
TIPSTER Architecture i
TIPSTER Text Program i
try 18, 22

U
Using TDM 9
Using the Exception Handling
Package 17