MERANT

COLLABORATION

QUALITY

Edition 2.3

MERANT® TRACKER™ TOOLKIT REFERENCE GUIDE

FROM THE MAKERS OF PVCS®

Copyright © 1992–2003. Merant All rights reserved.

Merant® and PVCS® are registered trademarks of Merant, and Merant Dimensions, Merant Mover, Merant Professional, Merant Meritage, Merant Tracker, Merant Version Manager, Merant Collage, Merant Build, Merant Modello, Merant Metrics, Merant Pulse, Merant TrackerLink, and PVCS Change Manager for Oracle are trademarks of Merant. All other trademarks are the property of their respective owners.

Portions of this software are copyrighted by DataDirect Technologies, 1991–2003.

No part of this publication may be copied, photocopied, reproduced, transmitted, transcribed, or reduced to any electronic medium or machine-readable form without prior written consent of Merant.

Licensee may copy the software product user documentation contained on a CD-ROM, but only to the extent necessary to support the users with authorized access to the software in accordance with the license agreement applicable to the software. Any reproduction of such software product user documentation, regardless of whether the documentation is reproduced in whole or in part, must be accompanied by this copyright statement in its entirety, without modification.

U.S. GOVERNMENT RESTRICTED RIGHTS. It is acknowledged that the Software and the Documentation were developed at private expense, that no part is in the public domain, and that the Software and Documentation are Commercial Computer Software provided with RESTRICTED RIGHTS under Federal Acquisition Regulations and agency supplements to them. Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in the license agreement accompanying such Software and Documentation and as provided in, among other sources, DFARS 252.227-7015 and 227.7202, or subparagraphs (c)(1) and (2) of the Commercial Computer Software Restricted Rights at FAR 52.227-19, as applicable. Contractor is Merant, 3445 NW 211th Terrace, Hillsboro, Oregon 97124. Rights are reserved under copyright laws of the United States with respect to unpublished portions of the Software.

Merant
3445 NW 211th Terrace
Hillsboro, Oregon 97124 U.S.A.

MA-TRKREF-001

Table of Contents

Welcome to Tracker 11

Licensing 11

Typographical Conventions 12

Ordering Hard-Copy Manuals 13

Contacting Technical Support 13

1 Introduction 15

Overview 16
- Components of the Toolkit 16
- Language Support 17

Calling Conventions 18
- Returned Data 18
- Data Types 19
- Considerations 20
- Accessing the Toolkit 20

2 Using the Tracker Toolkit 21

Using the Tracker Toolkit 22

Conventions 22
- Handles 23
- Project Login 25
- Lists and Sequential Access 28
- Record Transactions 30
- Importing and Exporting 32
- Callback Functions 36

Working with Information in the Tracker Database 38
- Records 38
- Issue Types 40

Notes 41
Attached Files 42
Module Associations 43

Sample Programs 44

3 Tracker Toolkit Functions 45

Tracker Toolkit Functions Listed By Category 46
Handle Functions 46
Project Login Functions 47
Record Functions 48
Attached File Functions 51
Note Functions 52
Module Association Functions 53
Import and Export Functions 55
Callback Functions 55
Issue Type Functions 56

Tracker Toolkit Functions Listed Alphabetically 57

TrkAddNewAssociation 58

TrkAddNewAttachedFile 59

TrkAddNewNote 60

TrkAssociationHandleAlloc 61

TrkAssociationHandleFree 62

TrkAttachedFileHandleAlloc 63

TrkAttachedFileHandleFree 64

TrkCloneRecordBegin 65

TrkCloneRecordCommit 66

TrkDeleteAssociation 67

TrkDeleteAttachedFile 68

TrkDeleteNote 69

TrkDeleteRecord 70

TrkExportHandleAlloc 71
TrkExportHandleFree. 72
TrkExportOneRecord 73
TrkExportRecordsBegin 74
TrkExportRecordsClose 77
TrkExtractAttachedFile 78
TrkGetAssociationModuleName 79
TrkGetAssociationRevisionFixed 80
TrkGetAssociationRevisionFound 81
TrkGetAssociationText 82
TrkGetAssociationTextLength 83
TrkGetAssociationTimeFixed 84
TrkGetAssociationTimeFound 85
TrkGetAssociationTransactionID 86
TrkGetAssociationUser. 87
TrkGetAttachedFileName 88
TrkGetAttachedFileStorageMode 89
TrkGetAttachedFileTime 90
TrkGetAttachedFileTransactionID 91
TrkGetDescriptionAccessRights 92
TrkGetDescriptionData 93
TrkGetDescriptionDataLength 94
TrkGetDescriptionTransactionID 95
TrkGetFieldAccessRights 96
TrkGetFieldDefaultNumericValue 97
TrkGetFieldDefaultStringValue 98
TrkGetFieldMaxDataLength 99

TrkGetFieldRange 101
TrkGetFieldTransactionID 103
TrkGetFieldType 104
TrkGetIniFile 106
TrkGetLoginDBMSName 107
TrkGetLoginDBMSType. 108
TrkGetLoginProjectName 109
TrkGetLoginUserName 110
TrkGetNextAssociation 111
TrkGetNextAttachedFile. 112
TrkGetNextChoice. 113
TrkGetNextDBMSType 114
TrkGetNextField 115
TrkGetNextIssueType 116
TrkGetNextNote 117
TrkGetNextProject 118
TrkGetNextQueryName 119
TrkGetNextRecord 120
TrkGetNextRecordType. 121
TrkGetNextUser 122
TrkGetNoteAuthor 123
TrkGetNoteCreateTime 124
TrkGetNoteData 125
TrkGetNoteDataLength 126
TrkGetNoteModifyTime 127
TrkGetNoteTitle 128
TrkGetNoteTransactionID. 129

TrkGetNumericAttribute 130
TrkGetNumericFieldValue 131
TrkGetProjectTransactionID. 132
TrkGetQueryRecordType 133
TrkGetRecordRecordType 134
TrkGetRecordTransactionID. 135
TrkGetRecordTypeName 136
TrkGetSingleRecord 137
TrkGetStringFieldValue 138
TrkGetUserEmail 139
TrkGetUserFullName 140
TrkHandleAlloc. 141
TrkHandleFree 142
TrkImportHandleAlloc 143
TrkImportHandleFree 144
TrkImportInit 145
TrkImportNewRecords. 148
TrkImportSetOptions 149
TrkInitAssociationList. 152
TrkInitAttachedFileList. 153
TrkInitChoiceList. 154
TrkInitDBMSTypeList 155
TrkInitFieldList 156
TrkInitIssueTypeList 157
TrkInitNoteList 158
TrkInitProjectList 159
TrkInitProjectListEx 161

TrkInitQueryNameList 162
TrkInitRecordTypeList 163
TrkInitUserList 164
TrkInTrayInitRecordList 165
TrkModifyIssueType 167
TrkNewRecordBegin 168
TrkNewRecordBeginEx 169
TrkNewRecordCommit 170
TrkNoteHandleAlloc 171
TrkNoteHandleFree 172
TrkProjectLogin 173
TrkProjectLoginEx 175
TrkProjectLogout 177
TrkQueryInitRecordList 178
TrkRecordCancelTransaction 180
TrkRecordHandleAlloc 181
TrkRecordHandleFree 182
TrkRegisterCallback 183
Callback Function Prototype 183
Callback Messages 184
TrkSetAssociationModuleName 189
TrkSetAssociationRevisionFixed 190
TrkSetAssociationRevisionFound 191
TrkSetAssociationText 192
TrkSetAssociationTimeFixed 193
TrkSetAssociationTimeFound 194
TrkSetAssociationUser 195

TrkSetDescriptionData 196
TrkSetIniFile 197
TrkSetNoteAuthor 198
TrkSetNoteCreateTime 199
TrkSetNoteData 200
TrkSetNoteModifyTime 201
TrkSetNoteTitle 202
TrkSetNumericAttribute 203
TrkSetNumericFieldValue 204
TrkSetStringFieldValue 205
TrkUnregisterCallback 206
TrkUpdateRecordBegin 207
TrkUpdateRecordCommit 208

4 Return Values 209

Return Values 210

Appendix A: Glossary of Tracker Toolkit Terms 215

Index 217

Welcome to Tracker

Thank you for choosing Merant® Tracker™ Administrator, the administration tool for Tracker.

Purpose of this manual

This manual contains conceptual and "how to" information on using Tracker Toolkit.

For more information

Refer to the *Tracker User's Guide* for a description of the Tracker documentation set, a summary of the ways to work with Tracker, and instructions for accessing the Online Help.

Edition status

This is Edition *2.3* of the *Tracker Toolkit Reference Guide.* The information in this edition applies to *Release 8.0 of Tracker* or later. This edition supersedes earlier editions of this manual.

Licensing

Each developer who uses the Tracker Toolkit must license one copy of Tracker. This license does not grant you permission to distribute applications you create with the Toolkit. If you distribute an application that calls Tracker functions, you must purchase a Tracker license for each end user of the application.

Typographical Conventions

The following typographical conventions are used in the online manuals and online help. These typographical conventions are used to assist you when using the documentation; they are not meant to contradict or change any standard use of typographical conventions in the various product components or the host operating system.

Convention	Explanation
italics	Introduces new terms that you may not be familiar with and occasionally indicates emphasis.
bold	Emphasizes important information and field names.
UPPERCASE	Indicates keys or key combinations that you can use. For example, press the ENTER key.
`monospace`	Indicates syntax examples, values that you specify, or results that you receive.
`monospaced italics`	Indicates names that are placeholders for values you specify; for example, *`filename`*.
`monospace bold`	Indicates the results of an executed command.
vertical rule |	Separates menus and their associated commands. For example, select File | Copy means to select Copy from the File menu. Also, indicates mutually exclusive choices in a command syntax line.
brackets []	Indicates optional items. For example, in the following statement: `SELECT [DISTINCT]`, DISTINCT is an optional keyword.
. . .	Indicates command arguments that can have more than one value.

Ordering Hard-Copy Manuals

As part of your Tracker license agreement, you may print and distribute as many copies of the Tracker manuals as needed.

If you do not want to print each of these online manuals, you can order hard-copy versions from Merant. To order, please contact your sales representative for assistance

Contacting Technical Support

Merant provides technical support for all registered users of this product, including limited installation support for the first 30 days. If you need support after that time, contact Merant using one of the methods listed in the *Installation Guide*, the *Tracker User's Guide* or the Online Help.

Technical support is available 24 hours a day, 7 days a week, with language-specific support available during local business hours. For all other hours, technical support is provided in English.

Support via the web, E-mail, and telephone

SupportNet Customers can report problems and ask questions on the SupportNet web page: `http://support.merant.com/`

To submit an issue, click the **Report a Problem** link and follow the instructions. You can also submit issues via E-mail or phone. Refer to the *Installation Guide*, *Getting Started Guide*, or Online help for a list of contact numbers, including numbers to call for local language support.

The SupportNet Web site contains up-to-date technical support information. Our SupportNet Community shares information via the Web, automatic E-mail notification, newsgroups, and regional user groups.

SupportNet Online is our global service network that provides access to valuable tools and information for an online community for users. SupportNet Online also includes a KnowledgeBase, which contains how-to information and allows you to search on keywords for technical bulletins. You can also download fix releases for your Merant Professional products.

1 Introduction

In This Chapter

For information about…	See page…
Overview	16
Calling Conventions	18

Overview

The Tracker Toolkit is an application programming interface (API) for programmers who want to develop applications that use Tracker services. You can use the functions in the Tracker Toolkit to build your own interface to Tracker projects, to import and export Tracker project information, or to add Tracker features to your own applications.

Using the Tracker Toolkit, you can:

- Retrieve the information in issues
- Retrieve user, project and login information
- Retrieve information about a Tracker installation
- Submit new issues
- Update existing issues
- Add, modify and retrieve notes, attached files and module associations
- Run Queries
- Import and Export records

Components of the Toolkit

The following components of the Tracker Toolkit are located in the MERANT\TRACKER\TOOLKIT directory:

- *Dynamic link libraries* (DLLs) containing Tracker function and constant declarations: TRKTOOLN.DLL, EXPDLLN.DLL
- *Import libraries* for linking applications that call Tracker functions: TRKTOOLN.LIB, EXPDLLN.LIB

- A *header file*, TRKTOOL.H, containing definitions and function prototypes.
- Sample programs and header files for C/C++, Delphi Pascal and Basic.

Language Support

The Tracker Toolkit Reference Guide documents C/C++ and Basic function declarations. You can, however, call Tracker functions from other languages that support calls to external DLLs. The following examples show typical function declarations in C/C++, Delphi Pascal and Microsoft Visual Basic. For more information, refer to the specific language documentation.

C/C++

```
TRK_UINT TrkProjectLogin(
   TRK_HANDLE trkHandle,/* Input */
   TRK_CSTR userName,/* Input (optional) */
   TRK_CSTR password,/* Input (optional) */
   TRK_CSTR projectName,/* Input (optional) */
   TRK_CSTR DBMSType,/* Input (optional) */
   TRK_CSTR DBMSName,/* Input (optional) */
   TRK_CSTR DBMSUserName,/* Input (optional) */
   TRK_CSTR DBMSPassword,/* Input (optional) */
   TRK_DBMS_LOGIN_MODE DBMSLoginMode)/* Input */
```

Delphi Pascal

Complete function declarations with function wrappers are included in TRKTOOL.PAS, the Pascal header file included with the sample programs on the Tracker disks or CD-ROM.

```
function TrkProjectLogin(
trkHandle:          TRK_HANDLE;                 // Input
userName:          TRK_CSTR;                  // Input (optional)
password:          TRK_CSTR;                  // Input (optional)
projectName:       TRK_CSTR;                  // Input (optional)
DBMSType:          TRK_CSTR;                  // Input (optional)
DBMSName:          TRK_CSTR;                  // Input (optional)
DBMSUserName:      TRK_CSTR;                  // Input (optional)
```

```
DBMSPassword:        TRK_CSTR;                    // Input (optional)
DBMSLoginMode:       TRK_DBMS_LOGIN_MODE           // Input
): TRKAPI; stdcall;
```

Visual Basic Function and constant declarations for Visual Basic are contained in TRKTOOL.BAS, a Basic header file included with the sample programs on the Tracker disks or CD-ROM.

```
Declare Function TrkProjectLogin& Lib "TRKTOOLN"
   (ByVal trkHandle&, ByVal userName$, ByVal password$,
   ByVal projectName$, ByVal DBMSType$, ByVal DBMSName$,
   ByVal DBMSUserName$, ByVal DBMSPassword$, DBMSLoginMode&)
```

Calling Conventions

All functions return an integer value of zero, represented by the constant `TRK_SUCCESS`, if successful. Non-zero values indicate that an error has occurred. See Chapter 4, "Return Values," on page 209, for descriptions of the error codes.

Returned Data

Many Tracker Toolkit functions return data by reference to the caller. There are two types of returned data:

- *Long integers (4-byte)*. Often long integers returned to the caller represent a special value intended to be passed to another function, like handles, or the values representing record and field types.
  Note: Signed and unsigned ints are stored in the same manner and can simply be cast to the appropriate type. There are no special Toolkit functions that return either a signed int or an unsigned int. See "TrkGetNumericFieldValue" on page 131.

- *Buffers*, which are arrays of bytes similar to text strings. All functions that store information in buffers take a length or size argument. The maximum buffer size, represented by the constant `TRK_MAX_STRING`, is defined in TRKTOOL.H. If the buffer you allocate is too small for the data returned by Tracker, the called function returns an error.

  All strings returned by Toolkit functions are null-terminated. Functions with parameters that are pointers to text strings treat a null pointer and a pointer to a null string as equivalent.

Data Types

The following table lists the data types used in the Tracker Toolkit. These data types are defined in TRKTOOL.H.

Tracker Toolkit Data Type	C/C++	Visual Basic
TRK_STR	char FAR*	String
TRK_CSTR	const char FAR*	String
TRK_UINT	unsigned long	Long
TRK_BOOL	unsigned long	Long
TRK_CHAR	unsigned long	Long
TRK_VERSION_ID_TYPE	unsigned long	Long
TRK_DBMS_LOGIN_MODE	unsigned long	Long
TRK_RECORD_TYPE	unsigned long	Long
TRK_TRANSACTION_ID	unsigned long	Long
TRK_FIELD_TYPE	unsigned long	Long
TRK_TIME	unsigned long	Long
TRK_CREATE_MODE	unsigned long	Long
TRK_ACCESS_MODE	unsigned long	Long

Tracker Toolkit Data Type	C/C++	Visual Basic
TRK_FILE_STORAGE_MODE	unsigned long	Long
TRK_LIST_LINK_ORDER	unsigned long	Long
TRK_EMB_QUOTE	unsigned long	Long
TRK_DATE_FORMAT	unsigned long	Long
TRK_TIME_FORMAT	unsigned long	Long
TRK_DATE_OPTION	unsigned long	Long
TRK_CHOICE_OPTION	unsigned long	Long
TRK_USER_OPTION	unsigned long	Long
TRK_NUMBER_OPTION	unsigned long	Long

Considerations

If you are writing applications in C/C++ or Pascal, use the **standard calling convention.**

Accessing the Toolkit

The calling application must be able to find TRKTOOLN.DLL and EXPDLLN.DLL. The dynamic link libraries must be in the same directory as the calling application or in a directory included in the PATH statement.

2 Using the Tracker Toolkit

In This Chapter

For information about…	See page…
Using the Tracker Toolkit	22
Conventions	22
Working with Information in the Tracker Database	38
Sample Programs	44

Using the Tracker Toolkit

This chapter provides information on using the Tracker Toolkit (TTK). You should read this chapter before you begin using Tracker functions in your own applications. You may also want to refer back to it if you have questions about working with the Toolkit.

This chapter contains:

- An explanation of the conventions used by the Tracker Toolkit
- Information on working with data in the Tracker database
- Information on the sample programs included with the Toolkit

Before you begin integrating Tracker features into your own programs, you should also be familiar with the Tracker application, and the *Tracker User's Guide* and *Tracker Administrator's Guide*. For additional information on the terms used in this manual, consult the Glossary on page 215.

Conventions

Before you begin using Tracker functions in your own applications, you should be familiar with the conventions used by the Tracker Toolkit. You must follow these conventions every time you want to include a Tracker feature in a custom program, or want to call functions that operate on information in a Tracker database.

The following list briefly describes the primary conventions used by the Toolkit:

- **Handle allocation.** You must obtain handles to perform operations on any data associated with Tracker.
- **Project login/logout.** Before you can access a project's data, you must use the Toolkit to log in to the project, just as you do when using the Tracker application.
- **Lists and sequential access.** To retrieve information from a Tracker database, you must create in-memory lists of data items, and then access the items in the lists sequentially.
- **Record transactions.** To submit new records or update existing records in the database, you must call the set of functions that make up a record transaction.

The following sections provide explanations and related code examples for each of the above conventions. Explanations of the high-level import and export functions provided by the Toolkit, and instructions for using callback functions are also included.

Handles

A handle is a value used internally to refer to state information about Toolkit operations. As a programmer, you cannot use a handle to access state data directly. You can only pass a handle to functions that access the information for you. You should never perform any external operation that alters the value of a handle.

The Tracker Toolkit uses the following handles:

- **Tracker Handles,** which refer to information about the state of operations using the Toolkit, including the project to which the caller is logged in, and other general information.
- **Record Handles**, **Note Handles**, **Attached File Handles and Association Handles**, which refer to information about the

state of operations on record, note, attached file or module association data.

- **Import Handles** and **Export Handles**, which refer to information about the current state of import and export operations.

Technique

You must obtain a Tracker Handle before you can call any other Tracker functions. A Tracker Handle is a required parameter for the functions that perform project login and logout, and for functions that access general information about Tracker installations and projects.

You must obtain other handles in order to perform operations on related data, or to use the import and export features. For example, a Record Handle is a required parameter for all functions that work with information associated with records, including functions that submit a new record or update an existing one. An Import Handle is a required parameter for the functions you use to import records.

The simplest way to use handles is to allocate all the handles you need when you begin operations with the Toolkit, and then to release them all at once when you are finished.

When you are finished conducting operations related to a particular handle, call the corresponding release function to free the handle and release the internal resources associated with it.

Using Multiple Handles

In performing operations with the Tracker Toolkit, you will rarely need to use more than one handle of any one type at the same time. For example, you will rarely need to obtain more than one Tracker Handle, or more than one Record Handle.

The only situation in which you must allocate more than one handle of a particular type is one in which you want to perform concurrent operations on more than one project, or you want to copy items from one in-memory list to another. Otherwise, one allocated handle of any given type is sufficient.

Examples

Functions that allocate handles return them by reference, as is shown in the following example.

```
TRK_HANDLE trkHandle;

/* Obtain a Tracker Handle */
TrkHandleAlloc(TRK_VERSION_ID, &trkHandle);
```

Once you have obtained a handle, you can pass it on as an argument to other Tracker functions. When you are finished with operations associated with the handle, call the corresponding release function as shown:

```
/* Release a Tracker Handle */
TrkHandleFree(&trkHandle);
```

When you release a handle, the value returned by reference is zero.

Project Login

You must use the Toolkit to log in to a project before you can access its associated data, just as you do when you use the Tracker application. When you log in, you must provide a valid user name and password, as defined in the Tracker Administrator application. If you are using Microsoft SQL, Sybase or Oracle for your database, you must provide a DBMS user name and password as well.

You must log in to a project before you call most Tracker functions; those functions that do not require login are described in the “Custom Login” section below.

Technique

Call the function **TrkProjectLoginEx** to log in to a Tracker project. When you are through performing operations related to the project, call **TrkProjectLogout**.

Examples

The following example shows a simple project login. The example assumes that you have obtained a valid Tracker Handle and the name of a server definition.

```
/* Log in to a Tracker Project using serverName */
TrkProjectLoginEx(
   trkHandle,           /* Valid Tracker Handle */
   "Admin",             /* User name */
   "bA4e3",             /* User password */
   "Learning Project",  /* Project name */
   "TestServer");       /* Name from Server Wizard*/
```

Another example, using TrkProjectLogin() instead of TrkProjectLoginEx(), is appropriate when you have all of the specific DBMS information. This was the method used in versions of Tracker prior to Tracker 7.0--it will still work, but the serverName method is recommended.

```
/* Log in to a Tracker Project */
TrkProjectLogin(
   trkHandle,      /* Valid Tracker Handle */
   "Admin",        /* User name */
   "bA4e3",        /* User password */
   "Learning Project",/* Project name */
   "SQL Server",   /* DBMS Type */
   "m:\\tracker\\db",/* DBMS Name */
   "agillies",     /* DBMS user name */
   "fH5g9",        /* DMBS password */
   TRK_USE_SPECIFIED_DBMS_LOGIN);/* DBMS Login Mode
   */
```

When you are finished with operations for any given project, you can log out as shown in the following example:

```
/* Log out of a Tracker Project */
TrkProjectLogout(trkHandle);
```

Custom Logins

The Tracker Toolkit contains a set of functions that you can use to create custom login procedures for the users of your applications. You can use these functions to perform the following tasks:

- ***Create login dialog boxes.*** You can call functions that list and access the projects and DBMS types known to Tracker before you call **TrkProjectLoginEx**. For example, you can use these functions to display available projects and associated DBMS types in a list box at login. Then, allow users to choose the project to which they want to log in.

  The functions that list and access projects and DBMS types are **TrkInitProjectListEx, TrkInitProjectList**, **TrkGetNextProject**, **TrkInitDBMSTypeList** and **TrkGetNextDBMSType**.

- ***Use default or pre-defined login information***. **TrkProjectLogin** and **TrkInitProjectList** allow you to use default or pre-defined login information when you log in to a project, or list the projects known to Tracker.

  If you specify that you want the Toolkit to read default login information, the function will use the information in the .INI file currently used by Tracker. You can call **TrkGetIniFile** to return the name of the .INI file currently used, or call **TrkSetIniFile** to set a new one. Both of these functions may be called before login.

  If you specify that you want the Toolkit to read pre-defined login information, the function will use information defined by the Tracker Administrator.

Retrieving Default Login Information

You can retrieve information concerning the current login by calling **TrkGetLoginUserName**, **TrkGetLoginProjectName**, **TrkGetLoginDBMSType**, or **TrkGetLoginDBMSName**.

Differences Between Tracker 7.x or Higher and Earlier Versions

In Tracker 7.x or higher, there is no "last login" entry to be read. That functionality is still present for Tracker 6.x users, but because of the login changes, is not available in Tracker 7.x or higher.

Lists and Sequential Access

The Tracker Toolkit provides functions that allow you to retrieve information about projects and the information in records, notes, attached files and module associations.

When you want to retrieve this information from a Tracker database, you must call functions that operate on the *list and sequential access* model. This model uses two kinds of functions:

- `List functions,` which create in-memory lists of items of a specific type found in a database. These items can be projects, users, records, attached files, or other types of information used by Tracker.
- *Sequential access functions,* which allow you to access the items in an in-memory list. The first time you call a sequential access function, the first item in its corresponding list becomes the *current item*. Each subsequent time you call the function, the next item in the list becomes current. If you call a sequential access function beyond the end of a list, the function returns a value of `TRK_E_END_OF_LIST`.

  The *current item* in the list is the item on which subsequent operations take effect. Many Tracker functions operate on the current item in an in-memory list.

Technique

Call the list function that corresponds to the type of data you want to access; then call the corresponding access function in sequence until the desired item in the list is current. Once the desired item is current, you may call other functions that operate on its associated data.

For example, if you want to access an attached file associated with the current record, you must call **TrkInitAttachedFileList** to create an in-memory list of attached files, and then call **TrkGetNextAttachedFile** in sequence until the desired attached file is current. Then, call other Tracker functions that operate on the attached file.

If you want to access every item in the list, or if you want to add a new item at the end of the list, test the return value of each sequential access function you call against `TRK_E_END_OF_LIST`.

Lists and item currency data are part of the state information managed by handles. Each handle refers to only one list and one current item at a time.

Example

The following example prints out a list of the users defined in the current project. The example assumes that you have obtained a valid Tracker Handle and that you have successfully logged in to a project.

```
char userName[TRK_MAX_STRING];

/* Create an in-memory list of Users */
TrkInitUserList(trkHandle);

/* Access the Users in the list in sequence, until end-of-list */
while (TrkGetNextUser(trkHandle, TRK_MAX_STRING, userName) !=
   TRK_E_END_OF_LIST)
   printf("%s\n", userName);
```

Retrieving Records

The Tracker Toolkit allows you to access the records in a database in the following ways:

- You can call **TrkInTrayInitRecordList** and **TrkGetNextRecord** to access the In Tray of the user presently logged in using your application.

- You can call **TrkQueryInitRecordList** and **TrkGetNextRecord** to execute a stored Query and to access the resulting in-memory list.
- You can call **TrkGetSingleRecord** to access a record whose record type and record ID are known.

Once you have made the desired record current, you can call other Tracker functions that operate on its associated data.

Record Transactions

A record transaction is a set of Tracker function calls that makes a change to a record in a Tracker database. When you want to submit a new record to the database, or update the information associated with an existing record, you must do so within the context of a record transaction.

A record transaction has the following characteristics:

- A call to a *begin function.* Begin functions initialize the record transaction. When you want to submit a new record, call **TrkNewRecordBegin**. When you want to update an existing record, call **TrkUpdateRecordBegin**.
- Calls to functions that modify the fields or description of the record, or add, delete or modify a note, an attached file or a module association.
- A call to a *commit function*. Commit functions send the information to the database and end the record transaction. Call **TrkNewRecordCommit** when you want to commit a new record to the database, and call **TrkUpdateRecordCommit** when you want to commit the changes you made to an existing record.

Technique

Call the appropriate begin function to initialize the record transaction. Then, make changes to fields and descriptions, or add, delete or modify notes, attached files or module associations. When you are finished making changes, call the appropriate commit function.

Example

The following example shows a simple procedure for updating an existing record. The example assumes that you have already obtained a valid Tracker Handle and that you have logged in to a project.

```
TRK_RECORD_HANDLE trkRecordHandle;
TRK_ATTFILE_HANDLE trkAttFileHandle;
TRK_TRANSACTION_ID transactionID;

/* Obtain Record and Attached file Handles */
TrkRecordHandleAlloc(trkHandle, &trkRecordHandle);
TrkAttachedFileHandleAlloc(trkRecordHandle, &trkAttFileHandle);

/* Create an in-memory list of the Records in the User's In Tray
*/
TrkInTrayInitRecordList(trkRecordHandle, recordType, 0, 0);

/* Access first Record in the in-memory list */
TrkGetNextRecord(trkRecordHandle);

/* Begin the Record transaction */
TrkUpdateRecordBegin(trkRecordHandle);

/* Modify the "Title" Field of the Record */
TrkSetStringFieldValue(trkRecordHandle, "Title", "New Login
Dialog");

/* Modify the Description of the Record */
TrkSetDescriptionData(trkRecordHandle, 255, "Update Description",
0);

/* Add a new Attached File */
TrkAddNewAttachedFile(trkAttFileHandle, "D:\\NEWFILE.TXT",
TRK_FILE_ASCII);

/* Commit updated Record to the database */
TrkUpdateRecordCommit(trkRecordHandle, &transactionID);
```

```
/* Release handles */
TrkAttachedFileHandleFree(&trkAttFileHandle);
TrkRecordHandleFree(&trkRecordHandle);
```

Considerations

Note the following considerations when you are submitting or updating records to the database:

- You may call **TrkRecordCancelTransaction** at any time before you commit the record transaction.
- If the logged in user does not have Read and Write access rights for any field or description modified during a record transaction, the commit function will return an error.
- You must make changes to a record, attached file, note or module association in the context of a record transaction. If you do not, the changes will not be made in the database.

Importing and Exporting

The Tracker Toolkit provides sets of high-level functions that allow you to import and export records to and from a text file.

You can only import files that have been previously exported from the Toolkit or from the Tracker application using Tracker format.

Exporting Records

A record export session consists of a set of Tracker function calls that send record information to a text file. To export one or more records, you must perform the following operations:

- *Obtain an Export Handle and initialize the export feature.* Call **TrkExportHandleAlloc** to obtain an Export Handle. Then, call **TrkExportRecordsBegin** to initialize the export feature and to specify the format and filename for the exported file.

You must also specify whether the records you wish to export are intended to be re-imported back into Tracker.

- *Make the record you want to export current, and export the record.* Call functions that list and access records to make the record you want to export current, and then call **TrkExportOneRecord** to export the record. Then make additional records current and export them to the same file as needed.
- *Close the export session and free the Export Handle.* Call **TrkExportRecordsClose** to close the text file and release the associated internal resources; then call **TrkExportHandleFree** to release the Export Handle.

Example

The following example shows a simple use of the record export feature to export the records in a user's In Tray to a file in Tracker format. The resulting file is then suitable for import back into the database. The example assumes that you have already obtained a Tracker handle and that you have logged in to a project.

```
TRK_RECORD_HANDLE trkRecordHandle;
TRK_EXPORT_HANDLE trkExportHandle;
TRK_BOOL true = 1;

/* Obtain Handles */
TrkRecordHandleAlloc(trkHandle, &trkRecordHandle);
TrkExportHandleAlloc(trkHandle, &trkExportHandle);
/* Initialize Record export feature */
TrkExportRecordsBegin(trkExportHandle,
   recordType,
   "D:\\EXPORT.TXT",
   true,
   0,
   0,
   0,
   0,
   0);

/* Create an in-memory list of the Records in the User's In Tray
*/
```

```
TrkInTrayInitRecordList(trkRecordHandle, recordType, 0, 0);

/* Access each Record in list until end-of-list, */
/* Exporting each Record to the text file as it becomes current */
while (TrkGetNextRecord(trkRecordHandle) == TRK_SUCCESS)
   TrkExportOneRecord(trkExportHandle, trkRecordHandle);

/* Close Record export feature and text file */
TrkExportRecordsClose(trkExportHandle);

/* Release Handles */
TrkExportHandleFree(&trkExportHandle);
TrkRecordHandleFree(&trkRecordHandle);
```

Importing Records

A record import session consists of a set of Tracker functions that retrieve record information from a text file. You may only import records that have been previously exported in Tracker format using the Tracker Toolkit or the Tracker application.

To import one or more records, you must perform the following operations:

- *Obtain an Import Handle and initialize the import feature.* Call **TrkImportHandleAlloc** to obtain an Import Handle. Then call **TrkImportInit** to initialize the import feature and to specify the name of the file from which you want to import records.
- *Set import options.* Call **TrkImportSetOptions** to set options that specify the action you want the Toolkit to take, should it encounter bad values in the text file. If you omit this step, the default values specified in the function description will be used.
- *Import records from the file.* Call **TrkImportNewRecords** to import the records. The fields being imported will automatically be mapped by type and name. Optionally, you

can specify the name of a file to which the Toolkit can send a log of the import operation.

- *Release the Import Handle.* Call **TrkImportHandleFree** to release the import handle and associated internal resources.

Example

The following example shows a simple use of the record import feature. The example assumes that you have already obtained a Tracker handle and that you have logged in to a project.

```
TRK_IMPORT_HANDLE trkImportHandle;
TRK_BOOL true = 1;

/* Obtain Import Handle */
TrkImportHandleAlloc(trkHandle, &trkImportHandle);

/* Initialize Record import feature */
TrkImportInit(trkImportHandle,
   "D:\\EXPORT.TXT",
   true,
   recordType
   0,
   0,
   0,
   0);

/* Set options for Record import */
/* Using options to set bad Choice, User, Number Fields to default
*/
/* Setting bad Date Fields to current date */
TrkImportSetOptions(trkImportHandle,
   TRK_DEFAULT_CHOICE,
   TRK_DEFAULT_USER,
   TRK_DEFAULT_NUMBER,
   TRK_SET_CURRENT,
   "");

/* Import Records from the text file */
TrkImportNewRecords(trkImportHandle, "D:\\ERRLOG.TXT");

/* Release Import Handle */
TrkImportHandleFree(&trkImportHandle);
```

Callback Functions

A callback function is a function of your design that is called by the Tracker Toolkit whenever specific events occur. You can use callback functions to log Tracker function calls, to handle certain errors, or to track progress on specific Toolkit operations.

Callback functions can handle the following events:

- An exit from a Tracker function call
- Tracker Handle de-allocation
- An ODBC error
- An invalid field value
- A forced logout
- In Tray record retrieval activity
- Query execution activity
- Import activity

Callback function prototype

The callback functions you write must use the callback function prototype defined by the Tracker Toolkit in TRKTOOL.H.

```
TRKAPI LOADDS myCallback(
   TRK_UINT msg,
   TRK_UINT param1,
   TRK_UINT param2,
   TRK_CSTR strParam,
   TRK_UINT userData)
```

When the Tracker Toolkit calls a callback function, it passes specific information to the function's parameters. This information includes a message that identifies the event that has occurred, and other information that varies according to the event.

For example, when a Tracker function exits, the Toolkit passes a value of `TRK_MSG_API_TRACE` to the *`msg`* parameter, the return

code of the exiting function to the *param1* parameter, and the name of the exiting function to the *strParam* parameter.

Example

The following example is of a callback function designed to log Tracker function calls. In this example, the function first tests for the appropriate callback message (TRK_MSG_API_TRACE). If the test succeeds, the function executes the desired action that corresponds to the event, and then returns a value of TRK_MSG_HANDLED. If the test does not succeed, the function performs no other action besides returning a TRK_MSG_NOT_HANDLED value.

```
TRKAPI LOADDS funcTrace(
   TRK_UINT msg,
   TRK_UINT param1,
   TRK_UINT param2,
   TRK_CSTR strParam,
   TRK_UINT userData)
{
   if (msg == TRK_MSG_API_TRACE){
      if (param1 != TRK_SUCCESS){
         /* Log errors */
         FILE* fp = (FILE*) userData;
         fprintf(fp, "API Function '%s' returned error code %ld\n",
         strParam, param1);
      }
      return TRK_MSG_HANDLED;
   }
   return TRK_MSG_NOT_HANDLED;
}
```

You must call **TrkRegisterCallback** to register your callback functions before they can be used by the Toolkit. Each function you register is added to a list of callback functions. You may use as many callback functions as you need. Each time the Tracker Toolkit encounters one of the events it allows to be handled with callback functions, it calls every function in the list of registered callbacks, until a value of TRK_MSG_HANDLED is returned by one of the functions.

See "TrkRegisterCallback" on page 183 for more information on registering callback functions and callback events.

Working with Information in the Tracker Database

Records

A record is the basic unit of information in a Tracker database. Tracker records are known as issues, which contain information used to track product defects and system changes.

Versions of Tracker prior to 7.0 also used a second type of record known as Time Records, which contained information about product testing.

Record types

Each type of record is represented by a record type value in the Tracker Toolkit. This value is a required parameter for many functions that operate on record data, or provide general information about queries or fields. To obtain this value, you can:

- Call **TrkInitRecordTypeList** to create an in-memory list of record types defined in the current project, and then call **TrkGetNextRecordType** to access the record type values.
- Call **TrkGetRecordRecordType** to retrieve the record type of the current record.
- Call **TrkGetQueryRecordType** to retrieve the record type associated with a Query defined in the current project.

Once you have obtained a record type value, you can call **TrkGetRecordTypeName** to retrieve the name of the record type.

Fields

The information in a record is organized in categories called fields. The Tracker Toolkit recognizes five field types: Choice, String, Numeric, Date and Elapsed Time. Three standard Choice

fields (Submitter, Owner and User) are also recognized. To list and access the fields defined in the current record, call **TrkInitFieldList** and **TrkGetNextField**.

Field Types

Each type of field is represented by a field type value in the Tracker Toolkit. Retrieve this value to find out the field type of a field before you operate on its associated data. To obtain this value, you can:

- Call **TrkInitFieldList** to create an in-memory list of fields in a specified type of record, and then call **TrkGetNextField** to access the names and types of individual fields.
- Call **TrkGetFieldType** to retrieve the field type of the given field in a record.

The following is a list of field types and the associated constants defined in TRKTOOL.H.

Field Type	Constant
Choice	`TRK_FIELD_TYPE_CHOICE`
String	`TRK_FIELD_TYPE_STRING`
Numeric	`TRK_FIELD_TYPE_NUMBER`
Date	`TRK_FIELD_TYPE_DATE`
Elapsed Time	`TRK_FIELD_TYPE_ELAPSED_TIME`
Submitter	`TRK_FIELD_TYPE_SUBMITTER`
Owner	`TRK_FIELD_TYPE_OWNER`
User	`TRK_FIELD_TYPE_USER`

General Information About Fields

The Tracker Toolkit also provides functions that allow you to retrieve general information about fields. You can call **TrkGetFieldMaxDataLength** to retrieve the maximum data length allowed for a field, or call **TrkGetFieldRange** to retrieve the minimum and maximum allowed values for a Numeric field. You can also call **TrkGetFieldDefaultNumericValue** or **TrkGetFieldDefaultStringValue** to obtain a field's default values.

Retrieving and Modifying Field Values

You can use Tracker functions to retrieve or modify the value of any field in a record. You can call **TrkGetNumericFieldValue** to retrieve the value of a Numeric field, or call **TrkGetStringFieldValue** to retrieve the value of a String, Choice or Date field. To modify the value of a Numeric field, call **TrkSetNumericFieldValue**. To modify the value of a String, Choice or Date field, call **TrkSetStringFieldValue**.

Descriptions

A description consists of text information that describes a record. Each record may have only one description. The Tracker Toolkit allows you to retrieve or modify a description's contents. You can also retrieve a Transaction ID for the description.

You can retrieve the contents of a description by calling **TrkGetDescriptionDataLength** to obtain the length of the description, and then calling **TrkGetDescriptionData**. To modify a description, call **TrkSetDescriptionData**.

To retrieve a description's Transaction ID, call **TrkGetDescriptionTransactionID**.

Issue Types

An issue type is a classification that groups together related records so they can be managed differently from other related groups of records. A Tracker Administrator can define multiple issue types in Tracker to be used with system level issue types for the purpose of creating relationships between issue shared across projects.

Retrieving and Setting Issue Type Values

You can retrieve the issue type of a record by treating an issue type like a field. The issue type field will be returned by

TrkInitFieldList/TrkGetNextField , and you can get the value by calling **TrkGetStringFieldValue**.

Trying to use **TrkSetStringFieldValue** to set the value returns the **TRK_E_SET_FIELD_DENIED** code.

Retrieving and Setting Values for Invalid Fields

Trying to set or retrieve a value for a field that is invalid for the current record type returns the **TRK_E_INVALID_FIELD** error code.

Notes

A note is additional text information a user attaches to a record. Each record may have multiple notes, or none at all. The Tracker Toolkit provides a set of functions that allow you to retrieve note information, modify a note, add new notes or delete a note entirely. To list and access the notes associated with the current record, call **TrkInitNoteList** and **TrkGetNextNote**.

You can attach a new note to the current record by calling **TrkAddNewNote**. Once the new note is added, you can modify its contents, or other information associated with it. To delete a note, call **TrkDeleteNote**.

Retrieving and Modifying Notes

You can retrieve or modify the following information associated with a note:

- *Title:* Call **TrkGetNoteTitle** to retrieve the Title of the current note, or call **TrkSetNoteTitle** to modify the Title of an existing note.
- *Author:* Call **TrkGetNoteAuthor** to retrieve the name of the user who created the current note, or call **TrkSetNoteAuthor** to modify the Author of an existing note.
- *Contents:* Call **TrkGetNoteDataLength** to obtain the length of the data contained by the current note, and then call **TrkGetNoteData** to retrieve the note's contents. Call **TrkSetNoteData** to modify the contents of an existing note.

- *Timestamps:* Call **TrkGetNoteCreateTime** to retrieve the timestamp representing the time the current note was created, or call **TrkGetNoteModifyTime** to retrieve a timestamp for the time the note was last modified. You can call **TrkSetNoteCreateTime** to set the create timestamp, or call **TrkSetNoteModifyTime** to set the modify timestamp.
- *Transaction ID:* Call **TrkGetNoteTransactionID** to retrieve the database event number that represents the transaction in which the note was created or most recently modified.

Attached Files

An attached file is a binary or ASCII file attached to a record. The Tracker Toolkit provides functions that allow you to add a new attached file to a record, to extract an attached file, or to delete an attached file entirely. You can also retrieve general information about an attached file. To list and access the attached files associated with a record, call **TrkInitAttachedFileList**, and then **TrkGetNextAttachedFile**.

Call **TrkAddNewAttachedFile** to attach a new file to the current record, and call **TrkDeleteAttachedFile** to delete an attached file. To extract an attached file, call **TrkExtractAttachedFile**.

You can retrieve the following general information about attached files:

- *Name:* Call **TrkGetAttachedFileName** to retrieve the name of an attached file. You can use this function, for example, to list the names of the attached files associated with a record, or to use as the new filename when you extract an attached file.
- *Storage Mode:* Call **TrkGetAttachedFileStorageMode** to retrieve the storage mode of an attached file before you call **TrkExtractAttachedFile**.
- *Timestamp:* Call **TrkGetAttachedFileTime** to retrieve the timestamp for the time a file was attached to its record.

- *TransactionID:* Call **TrkGetAttachedFileTransactionID** to retrieve the database event number for the transaction during which a file was attached to its record.

Module Associations

You can use the set of Tracker functions that operate on module association data to write applications that integrate Tracker and Merant Version Manager features. Before you use these functions you should be familiar with the Tracker User's Guide, Chapter 9, "*Integrating Tracker with Version Control*" You should also be familiar with the Version Manager application.

Tracker functions that work with Version Manager allow you to add a new module association, retrieve module association information, modify the contents of a module association, or delete a module association entirely. To list and access the module associations linked to a record, call **TrkInitAssociationList**; then call **TrkGetNextAssociation**.

You can add a new module association to an issue by calling **TrkAddNewAssociation**. Once you have added a module association, you can call functions that modify the information associated with it. Call **TrkDeleteAssociation** to delete a module association entirely.

Retrieving and Modifying Module Associations

You can retrieve or modify the following information associated with a module association:

- *Change description:* Call **TrkGetAssociationTextLength** and **TrkGetAssociationText** to retrieve the change description for the current module association. Call **TrkSetAssociationText** to modify the change description.
- *Module Name:* Call **TrkGetAssociationModuleName** to retrieve the name of the Module linked to the issue by the current module association. Call

TrkSetAssociationModuleName to set the name of the Module you want linked by the module association.

- *User:* Call **TrkGetAssociationUser** to retrieve the name of the user who created the current module association. Call **TrkSetAssociationUser** to set the name of the user.
- *Revision Found:* Call **TrkGetAssociationRevisionFound** to retrieve the revision number or version label for the revision at which the defect reported by the issue was found. Call **TrkSetAssociationRevisionFound** to set the revision number or version label.
- *Revision Fixed:* Call ***TrkGetAssociationRevisionFixed*** to retrieve the revision number or version label for the revision at which the defect reported by the issue was fixed. Call **TrkSetAssociationRevisionFixed** to set the revision number or version label.
- *Time found:* Call **TrkGetAssociationTimeFound** to retrieve the timestamp for the time at which the defect reported by the issue was found. Call **TrkSetAssociationTimeFound** to set the timestamp.
- *Time fixed:* Call **TrkGetAssociationTimeFixed** to retrieve the timestamp for the time when the defect reported by the issue was fixed. Call **TrkSetAssociationTimeFixed** to set the timestamp.

Sample Programs

The Tracker Toolkit includes several sample programs written in C/C++, Delphi Pascal and Basic. This programs are located in the MERANT/TRACKER/TTK/EXAMPLES directory of your Tracker installation. Each program is accompanied by a README file that contains a program description and additional information.

3 Tracker Toolkit Functions

In This Chapter

For information about…	See page…
Tracker Toolkit Functions Listed By Category	46
Tracker Toolkit Functions Listed Alphabetically	57

Tracker Toolkit Functions Listed By Category

If you need to access a particular function in the Toolkit but you do not know the function name, refer to the lists of functions in this section. When you find the function you need, locate the full description in the alphabetical reference section of this chapter.

This section contains lists of Tracker Toolkit functions organized in the following categories:

- Handle Functions
- Project Login Functions
- Record Functions
- Attached File Functions
- Note Functions
- Module Association Functions
- Import and Export Functions
- Callback Functions
- Issue Type Functions

Handle Functions

Call handle functions to obtain and release any handles you need to perform Toolkit operations.

Call this function ...	To ...
TrkAssociationHandleAlloc	Obtain an Association Handle
TrkAssociationHandleFree	Release an Association Handle
TrkAttachedFileHandleAlloc	Obtain an Attached File Handle
TrkAttachedFileHandleFree	Release an Attached File Handle

Call this function ...	To ...
TrkExportHandleAlloc	Obtain an Export Handle
TrkExportHandleFree	Release an Export Handle
TrkHandleAlloc	Obtain a Tracker Handle
TrkHandleFree	Release a Tracker Handle
TrkImportHandleAlloc	Obtain an Import Handle
TrkImportHandleFree	Release an Import Handle
TrkNoteHandleAlloc	Obtain a Note Handle
TrkNoteHandleFree	Release a Note Handle
TrkRecordHandleAlloc	Obtain a Record Handle
TrkRecordHandleFree	Release a Record Handle

Project Login Functions

Call project login functions to log in and out of projects, to obtain information about project logins, and to prepare custom login procedures.

Call this function ...	To ...
TrkGetIniFile	Obtain the name of the .INI file currently used by Tracker
TrkGetLoginDBMSName	Retrieve the DBMS name of the project to which the user is currently logged in
TrkGetLoginDBMSType	Retrieve the DBMS type of the project to which the user is currently logged in
TrkGetLoginProjectName	Retrieve the name of the project to which the user is currently logged in
TrkGetLoginUserName	Retrieve the name of the currently logged in user
TrkGetNextDBMSType	Access an in-memory list of DBMS types
TrkGetNextProject	Access an in-memory list of projects
TrkGetNextUser	Access an in-memory list of users

Call this function ...	To ...
TrkGetProjectTransactionID	Retrieve the last allocated Transaction ID for a record of the specified type in the project to which the user is currently logged in
TrkGetUserEmail	Retrieve the e-mail address of the current user
TrkGetUserFullName	Retrieve the full name of the current user
TrkInitDBMSTypeList	Create an in-memory list of DBMS types known to Tracker
TrkInitProjectList	Create an in-memory list of projects known to Tracker
TrkInitProjectListEx	Create an in-memory list of the projects associated with a server name
TrkInitUserList	Create an in-memory list of users defined in the current project
TrkProjectLogin	Log in to a project
TrkProjectLoginEx	Log in to a project using a servername
TrkProjectLogout	Log out of a project
TrkSetIniFile	Set a new .INI file

Record Functions

Call record functions to retrieve information about records and queries, to access records in a project, and to update and submit records to the database.

Call this function ...	To ...
TrkCloneRecordBegin	Begin a record transaction that submits a cloned record to the database

Call this function ...	To ...
TrkCloneRecordCommit	Submit a newly cloned record to the database and end the record transaction
TrkDeleteRecord	Delete the current record from an in-memory list of records
TrkGetDescriptionAccessRights	Retrieve access rights for a record's description
TrkGetDescriptionData	Retrieve a record's description
TrkGetDescriptionDataLength	Retrieve the length of a record's description
TrkGetDescriptionTransactionID	Retrieve a description's transaction ID
TrkGetFieldAccessRights	Retrieve access rights for a field
TrkGetFieldDefaultNumericValue	Retrieve a field's default numeric value
TrkGetFieldDefaultStringValue	Retrieve a field's default string value
TrkGetFieldMaxDataLength	Retrieve the maximum allowed data length of a field
TrkGetFieldRange	Retrieve the minimum and maximum values for a numeric field
TrkGetFieldTransactionID	Retrieve the database event number for the transaction in which the most recent change was made to the specified field in the current record
TrkGetFieldType	Retrieve the field type of a field
TrkGetNextChoice	Retrieve the next choice in an in-memory Choice list
TrkGetNextField	Access an in-memory list of fields
TrkGetNextQueryName	Retrieve the next query in an in-memory Query list

Call this function ...	To ...
TrkGetNextRecord	Access an in-memory list of records
TrkGetNextRecordType	Access an in-memory list of record types
TrkGetNumericAttribute	Retrieve the numeric value of an attribute
TrkGetNumericFieldValue	Retrieve the numeric value of a field
TrkGetQueryRecordType	Retrieve the record type associated with a query
TrkGetRecordRecordType	Retrieve the record type of the current record
TrkGetRecordTransactionID	Retrieve the database event numbers for the transactions in which the current record was submitted and most recently updated
TrkGetRecordTypeName	Retrieve the name of a record type
TrkGetSingleRecord	Access one record by record type and record ID
TrkGetStringFieldValue	Retrieve the string value of a field
TrkInitChoiceList	Initialize an in-memory list of the Choices associated with the specified Choice field
TrkInitFieldList	Create an in-memory list of fields defined for the current record
TrkInitQueryNameList	Initializes an in-memory list of the stored Queries defined in the current project
TrkInitRecordTypeList	Create an in-memory list of the record types defined in the current project
TrkInTrayInitRecordList	Create an in-memory list of the records in the user's In Tray

Call this function ...	To ...
TrkModifyIssueType	Modify the issue type of the current record
TrkNewRecordBegin	Begin a record transaction to submit a new record to the database
TrkNewRecordBeginEx	Begin a record transaction to submit a new record of a specific issue type to the database
TrkNewRecordCommit	Submit a new record to the database and end the record transaction
TrkQueryInitRecordList	Create an in-memory list of records resulting from the execution of a query
TrkRecordCancelTransaction	Cancel an update or submit record transaction
TrkSetDescriptionData	Set a record's description
TrkSetNumericAttribute	Set the numeric value of an attribute
TrkSetNumericFieldValue	Set the numeric value of a field
TrkSetStringFieldValue	Set the string value of a field
TrkUpdateRecordBegin	Begin an update record transaction
TrkUpdateRecordCommit	Update a record and end the record transaction

Attached File Functions

Call these functions to add, delete or extract an attached file, or to retrieve information related to attached files.

Call this function ...	To ...
TrkAddNewAttachedFile	Add a new attached file

Call this function ...	To ...
TrkDeleteAttachedFile	Delete an attached file
TrkExtractAttachedFile	Extract an attached file
TrkGetAttachedFileName	Retrieve the file name of an attached file
TrkGetAttachedFileStorageMode	Retrieve the storage mode of an attached file
TrkGetAttachedFileTime	Retrieve the timestamp of an attached file
TrkGetNextAttachedFile	Access an in-memory list of attached files
TrkInitAttachedFileList	Create an in-memory list of attached files associated with the current record

Note Functions

Call these functions to add, delete or modify a note, or to retrieve note information.

Call this function ...	To ...
TrkAddNewNote	Add a new note
TrkDeleteNote	Delete a note
TrkGetNextNote	Access an in-memory list of notes
TrkGetNoteAuthor	Retrieve the name of a note's author
TrkGetNoteCreateTime	Retrieve the create timestamp of a note
TrkGetNoteData	Retrieve note data
TrkGetNoteDataLength	Retrieve the length of a note
TrkGetNoteModifyTime	Retrieve the modify timestamp of a note
TrkGetNoteTitle	Retrieve the title of a note

Call this function ...	To ...
TrkGetNoteTransactionID	Retrieve a note's transaction ID
TrkInitNoteList	Create an in-memory list of notes
TrkSetNoteAuthor	Set the name of a note's author
TrkSetNoteCreateTime	Set the create timestamp of a note
TrkSetNoteData	Set the contents of a note
TrkSetNoteModifyTime	Set the modify timestamp of a note
TrkSetNoteTitle	Set the title of a note

Module Association Functions

Call these functions to add, delete or modify module associations, or to retrieve module association information.

Call this function ...	To ...
TrkAddNewAssociation	Add a new module association
TrkDeleteAssociation	Delete a module association
TrkGetAssociationModuleName	Retrieve the name of the module linked to a record by a module association
TrkGetAssociationRevisionFixed	Retrieve the revision number or version label at which a defect was fixed
TrkGetAssociationRevisionFound	Retrieve the revision number of version label at which a defect was found
TrkGetAssociationText	Retrieve the change description of a module association
TrkGetAssociationTextLength	Retrieve the length of the change description of a module association

Call this function ...	To ...
TrkGetAssociationTimeFixed	Retrieve the timestamp for the time at which a defect was fixed
TrkGetAssociationTimeFound	Retrieve the timestamp for the time at which a defect was found
TrkGetAssociationTransactionID	Retrieve the database event number for the transaction in which the current module association was created
TrkGetAssociationTransactionID	Retrieve the transaction ID of a module association
TrkGetAssociationUser	Retrieve the name of the user who created a module association
TrkGetNextAssociation	Access an in-memory list of module associations
TrkInitAssociationList	Create an in-memory list of module associations
TrkSetAssociationModuleName	Set the name of the module linked to a record by a module association
TrkSetAssociationRevisionFixed	Set the revision number or version label at which a defect was fixed
TrkSetAssociationRevisionFound	Set the revision number or version label at which a defect was reported
TrkSetAssociationText	Set the change description of a module association
TrkSetAssociationTimeFixed	Set the timestamp for the time at which a defect was fixed
TrkSetAssociationTimeFound	Set the timestamp for the time at which a defect was reported
TrkSetAssociationUser	Set the name of the user who created a module association

Import and Export Functions

Call these functions to import and export records to and from text files.

Call this function ...	To ...
TrkExportHandleAlloc	Obtain an Export Handle
TrkExportHandleFree	Release an Export Handle
TrkExportOneRecord	Export one record to a text file
TrkExportRecordsBegin	Initialize an export session
TrkExportRecordsClose	Close an export session
TrkImportHandleAlloc	Obtain an Import Handle
TrkImportHandleFree	Release an Import Handle
TrkImportInit	Initialize an import session
TrkImportNewRecords	Import records from a text file
TrkImportSetOptions	Set import options

Callback Functions

Call these functions to register or unregister callback functions.

Call this function ...	To ...
TrkRegisterCallback	Register a callback function for use in event-handling
TrkUnregisterCallback	Unregister a callback function

Issue Type Functions

Call these functions to work with a list of the issue types in the current project.

Call this function ...	To ...
TrkInitIssueTypeList	Initialize an in-memory list of the issue types defined in the current project
TrkGetNextIssueType	Gain sequential access to the issue types in an in-memory issue-type list

Tracker Toolkit Functions Listed Alphabetically

This section contains complete descriptions of the functions in the Tracker Toolkit. Functions are listed alphabetically.

The function descriptions contain the following elements:

- *A brief description of the function.* Additional information on using individual functions can be found in Chapter 2 of this .
- *A C/C++ function declaration.*
- *A Basic function declaration.* The Basic function declarations in this manual call the 32-bit libraries, TRKTOOLN.DLL and EXPDLLN.DLL.Complete Basic function and constant declarations are contained in the files TRKTOOL.BAS and EXPDLL.BAS, which are included with the sample programs on the Tracker disks or CD-ROM.
- *Parameters.* The parameter descriptions are for C/C++. Visual Basic programmers must substitute a string variable for a pointer to a text buffer and a long integer variable for a pointer to an integer buffer.
- *Related functions and related topics.* You can find additional information on each function, or related functions, by following these references.

TrkAddNewAssociation

This function adds a new module association at the current position of an in-memory module association list. The new module association then becomes the current module association.

C/C++

```
TRK_UINT TrkAddNewAssociation(
    TRK_ASSOC_HANDLE trkAssociationHandle)/* Input */
```

Basic

```
Declare Function TrkAddNewAssociation& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.

Related Functions

TrkDeleteAssociation

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkAddNewAttachedFile

This function adds a new attached file at the current position of an in-memory attached file list. The new file then becomes the current attached file.

C/C++

```
TRK_UINT TrkAddNewAttachedFile(
     TRK_ATTFILE_HANDLE trkAttFileHandle,/* Input */
     TRK_CSTR filename,/* Input */
     TRK_FILE_STORAGE_MODE storageMode)/* Input */
```

Basic

```
Declare Function TrkAddNewAttachedFile& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&, ByVal filename$, ByVal storageMode&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.
filename	Pointer to a string containing the name and path of the file you want to attach to the record.
storageMode	Integer that specifies the data storage mode for the attached file. Possible values are `TRK_FILE_BINARY` and `TRK_FILE_ASCII`.

Related Functions **TrkDeleteAttachedFile**

Related Topics For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, attaching
Tracker Toolkit Reference Guide	Attached Files

TrkAddNewNote

This function adds a new note at the current position of an in-memory note list. The new note then becomes the current note.

C/C++

```
TRK_UINT TrkAddNewNote(
    TRK_NOTE_HANDLE trkNoteHandle)/* Input */
```

Basic

```
Declare Function TrkAddNewNote& Lib "TRKTOOLN"
     (ByVal trkNoteHandle&)
```

Parameters

trkNoteHandle Handle returned by **TrkNoteHandleAlloc**.

Related Functions

TrkSetNoteAuthor
TrkSetNoteCreateTime
TrkSetNoteData
TrkDeleteNote

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes, attaching
Tracker Toolkit Reference Guide	Notes

TrkAssociationHandleAlloc

This function allocates an Association Handle, which is a required parameter for all Tracker Functions that operate on module association data. The handle you obtain remains open until you call **TrkAssociationHandleFree**.

C/C++

```
TRK_UINT TrkAssociationHandleAlloc(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_ASSOC_HANDLE FAR *pTrkAssociationHandle)/* Output */
```

Basic

```
Declare Function TrkAssociationHandleAlloc& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pTrkAssociationHandle&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pTrkAssociationHandle	Pointer to the variable that is to receive the Association Handle.

Related Functions

TrkAssociationHandleFree

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles, Module Associations

TrkAssociationHandleFree

This function closes the specified Association Handle and releases all memory associated with it. Call this function when you have completed all operations related to the Association Handle.

C/C++

```
TRK_UINT TrkAssociationHandleFree(
    TRK_ASSOC_HANDLE pTrkAssociationHandle)/* Input/Output */
```

Basic

```
Declare Function TrkAssociationHandleFree& Lib "TRKTOOLN"
    (pTrkAssociationHandle&)
```

Parameters

pTrkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.

Related Functions

TrkAssociationHandleAlloc

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles, Module Associations

TrkAttachedFileHandleAlloc

This function allocates an Attached File Handle, which is a required parameter for all Tracker functions that operate on attached file data. The handle you obtain remains open until you call **TrkAttachedFileHandleFree**.

C/C++

```
TRK_UINT TrkAttachedFileHandleAlloc(
    TRK_RECORD_HANDLE trkRecordHandle/* Input */
    TRK_ATTFILE_HANDLE FAR *pTrkAttFileHandle)/* Output */
```

Basic

```
Declare Function TrkAttachedFileHandleAlloc& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pTrkAttFileHandle&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pTrkAttFileHandle	Pointer to the variable that is to receive the Attached File Handle.

Related Functions

TrkAttachedFileHandleFree

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, attaching
Tracker Toolkit Reference Guide	Handles

TrkAttachedFileHandleFree

This function closes the specified Attached File Handle and releases all memory associated with it. Call this function when you have completed all operations related to the Attached File Handle.

C/C++

```
TRK_UINT TrkAttachedFileHandleFree(
    TRK_ATTFILE_HANDLE *pTrkAttFileHandle)/* Input/Output */
```

Basic

```
Declare Function TrkAttachedFileHandleFree& Lib "TRKTOOLN"
    (pTrkAttFileHandle&)
```

Parameters

pTrkAttFileHandle Handle returned by **TrkAttachedFileHandleAlloc**.

Related Functions

TrkAttachedFileHandleAlloc

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, attaching
Tracker Toolkit Reference Guide	Handles

TrkCloneRecordBegin

This function clears the current in-memory record and sets its fields to the original record's vurrent values or the default field values. Call this function to begin a record transaction that submits a cloned record to the database. When you are finished setting field values or other data associated with the record, call **TrkCloneRecordCommit**.

C/C++

```
TRK_UINT TrkCloneRecordBegin(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_RECORD_TYPE recordType)/* Input */
    TRK_RECORD_Handle trkOriginal)/* Input */
```

Basic

```
Declare Function TrkCloneRecordBegin& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal recordType&, ByVal trkOriginal&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
recordType	Integer value that represents the type of record you want to submit.
trkOriginal	Handle representing the original record being cloned.

Related Functions

TrkNewRecordBegin
TrkCloneRecordCommit

TrkCloneRecordCommit

This function submits a newly cloned record to the database and ends the record transaction. It also returns a Transaction ID representing the database event number for the submit transaction. You must call **TrkCloneRecordBegin** before you can use this function.

C/C++

```
TRK_UINT TrkCloneRecordCommit(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_TRANSACTION_ID FAR *pNewTransactionID)/* Output */
```

Basic

```
Declare Function TrkNewRecordCommit& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pNewTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pNewTransactionID	Pointer to the buffer that is to receive the new Transaction ID.

Related Functions

TrkNewRecordCommit
TrkCloneRecordBegin

TrkDeleteAssociation

This function deletes the current module association from an in-memory module association list.

C/C++

```
TRK_UINT TrkDeleteAssociation(
    TRK_ASSOC_HANDLE trkAssociationHandle)/* Input */
```

Basic

```
Declare Function TrkDeleteAssociation& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.

Related Functions

TrkAddNewAssociation

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkDeleteAttachedFile

This function deletes the current attached file from an in-memory attached file list.

C/C++

```
TRK_UINT TrkDeleteAttachedFile(
    TRK_ATTFILE_HANDLE trkAttFileHandle)/* Input */
```

Basic

```
Delcare Function TrkDeleteAttachedFile& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.

Related Functions

TrkAddNewAttachedFile

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, deleting
Tracker Toolkit Reference Guide	Attached Files

TrkDeleteNote

This function deletes the current note from an in-memory note list.

C/C++

```
TRK_UINT TrkDeleteNote(
    TRK_NOTE_HANDLE trkNoteHandle)/* Input */
```

Basic

```
Declare Function TrkDeleteNote& Lib "TRKTOOLN"
     (ByVal trkNoteHandle&)
```

Parameters

trkNoteHandle Handle returned by **TrkNoteHandleAlloc**.

Related Functions

TrkAddNewNote

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes, deleting
Tracker Toolkit Reference Guide	Notes

TrkDeleteRecord

This function deletes the current record from an in-memory list of records.

C/C++

```
TRK_UINT TrkDeleteRecord(
    TRK_RECORD_HANDLE trkRecordHandle)/* Input */
```

Basic

```
Declare Function TrkDeleteRecord& Lib "TRKTOOLN"
     (ByVal trkRecordHandle&)
```

Parameters

trkRecordHandle Handle returned by **TrkRecordHandleAlloc**.

Related Functions

TrkNewRecordBegin
TrkNewRecordSubmit

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Record Transactions

TrkExportHandleAlloc

This function allocates an Export Handle, which is a required parameter for all Tracker export functions. The handle you obtain remains open until you call **TrkExportHandleFree**.

C/C++

```
TRK_UINT TrkExportHandleAlloc(
    TRK_HANDLE trkHandle, /* Input */
    TRK_EXPORT_HANDLE FAR *pTrkExportHandle)/* Output */
```

Basic

```
Declare Function TrkExportHandleAlloc& Lib "EXPDLLW"
    (ByVal trkHandle&, pTrkExportHandle&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
pTrkExportHandle	Pointer to the variable that is to receive the Export Handle.

Related Functions

TrkExportHandleFree

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Exporting Records
Tracker Toolkit Reference Guide	Handles

TrkExportHandleFree

This function closes the specified Export Handle and releases all memory associated with it. Call this function when you have completed all operations related to the Export Handle.

C/C++

```
TRK_UINT TrkExportHandleFree(
    TRK_EXPORT_HANDLE FAR *pTrkExportHandle)/*Input/Output */
```

Basic

```
Declare Function TrkExportHandleFree& Lib "EXPDLLW"
    (pTrkExportHandle&)
```

Parameters

pTrkExportHandle Handle returned by **TrkExportHandleAlloc**.

Related Functions

TrkExportHandleAlloc

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Exporting Records
Tracker Toolkit Reference Guide	Handles

TrkExportOneRecord

This function exports the information in the current record to a text file. Before you use this function you must call **TrkExportRecordsBegin** to initialize the export feature and to specify a format for the exported data. When you have finished exporting, call **TrkExportRecordsClose**.

C/C++

```
TRK_UINT TrkExportOneRecord(
    TRK_EXPORT_HANDLE trkExportHandle,/* Input */
    TRK_RECORD_HANDLE trkRecordHandle)/* Input */
```

Basic

```
Declare Function TrkExportOneRecord& Lib "EXPDLLW"
    (ByVal trkExportHandle&, ByVal trkRecordHandle&)
```

Parameters

trkExportHandle	Handle returned by **TrkExportHandleAlloc**.
trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.

Related Functions

TrkExportRecordsBegin
TrkExportRecordsClose
TrkImportInit
TrkImportSetOptions
TrkImportNewRecords

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Exporting Records
Tracker Toolkit Reference Guide	Importing and Exporting

TrkExportRecordsBegin

This function initializes the record export feature and allows you to set a file name and format specifications for the exported information. To export record data, call **TrkExportOneRecord** after you call this function. When you are finished exporting, call **TrkExportRecordsClose**.

C/C++

```
TRK_UINT TrkExportRecordsBegin(
    TRK_EXPORT_HANDLE trkExportHandle,/* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_CSTR fileToWriteTo,/* Input */
    TRK_BOOL trackerFormat,/* Input */
    TRK_CHAR delimiterCharacter,/* Input (optional) */
    TRK_CHAR separatorCharacter,/* Input (optional) */
    TRK_EMB_QUOTE embeddedQuoteType,/* Input (optional) */
    TRK_DATE_FORMAT dateFormat,/* Input (optional) */
    TRK_TIME_FORMAT timeFormat)/* Input (optional) */
```

Basic

```
Declare Function TrkExportRecordsBegin& Lib "EXPDLLW"
    (ByVal trkExportHandle&, ByVal recordType&, ByVal fileToWriteTo$,
    ByVal trackerFormat&, ByVal delimiterCharacter&,
    ByVal separatorCharacter&, ByVal embeddedQuoteType&,
    ByVal dateFormat&, ByVal timeFormat&)
```

Parameters

trkExportHandle	Handle returned by **TrkExportHandleAlloc**.
recordType	Integer value that represents the record type of the record you want to export.
fileToWriteTo	Pointer to a string that contains the name and path of the file to which you want to export the record data.
trackerFormat	Possible values are `TRUE` and `FALSE`. You must set this parameter to `TRUE` if you plan to import the record data back into Tracker. Set this parameter to `FALSE` if you want to import the record into another application. If you set the value to `TRUE`, the remaining parameters will be ignored.

delimiterCharacter
: ASCII value of the character you want to use to mark the beginning and end of each field in the text file. This parameter is not required if *trackerFormat* is TRUE.

separatorCharacter
: ASCII value of the character you want to use to separate fields in the text file. This parameter is not required if *trackerFormat* is TRUE.

embeddedQuoteType
: Integer value representing the type of embedded quote you want to use in the resulting text file. Possible values are TRK_DOUBLE_QUOTE ("") and TRK_BACKSLASH_QUOTE (/"). This parameter is not required if *trackerFormat* is TRUE.

dateFormat
: Integer value representing the date format you want to use in the resulting text file. Possible values are TRK_CONTROL_PANEL_DATE and TRK_DBASE_FORMAT.

  The control panel date format is the default Windows date format. The dBase format is YYYYMMDD.

  This parameter is not required if *trackerFormat* is TRUE.

timeFormat
: Integer value representing the time format you want to use in the resulting text file. Possible values are TRK_CONTROL_PANEL_TIME (the default Windows format), TRK_24HOUR and TRK_24HOUR_LEADING_ZERO. This parameter is not required if *trackerFormat* is TRUE.

Related Functions

TrkExportOneRecord
TrkExportRecordsClose
TrkImportInit
TrkImportSetOptions
TrkImportNewRecords

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Exporting Records
Tracker Toolkit Reference Guide	Importing and Exporting

TrkExportRecordsClose

This function releases internal resources associated with a previous call to **TrkExportRecordsBegin** and closes the text file to which you exported record information.

C/C++

```
TRK_UINT TrkExportRecordsClose(
    TRK_EXPORT_HANDLE trkExportHandle)/* Input */
```

Basic

```
Declare Function TrkExportRecordsClose& Lib "EXPDLLW"
    (ByVal trkExportHandle&)
```

Parameters

trkExportHandle Handle returned by **TrkExportHandleAlloc**.

Related Functions

TrkExportRecordsBegin
TrkExportOneRecord

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Exporting Records
Tracker Toolkit Reference Guide	Importing and Exporting

TrkExtractAttachedFile

This function extracts the data contained in the current attached file. The file contents are copied into the file specified by the *filename* parameter. If a file by that name already exists, it is overwritten.

C/C++

```
TRK_UINT TrkExtractAttachedFile(
    TRK_ATTFILE_HANDLE trkAttFileHandle,/* Input */
    TRK_CSTR filename)    /* Input */
```

Basic

```
Declare Function TrkExtractAttachedFile& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&, ByVal filename$)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.
filename	Pointer to a string containing the name and path of the file to which you want to copy the contents of the attached file.

Related Functions

TrkInitAttachedFileList
TrkGetNextAttachedFile

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, attaching
Tracker Toolkit Reference Guide	Attached Files

TrkGetAssociationModuleName

This function returns the name of the Module linked to an issue by the current module association.

C/C++

```
TRK_UINT TrkGetAssociationModuleName(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR moduleName)   /* Output */
```

Basic

```
Declare Function TrkGetAssociationModuleName& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal bufferSize&,
    ByVal moduleName$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
bufferSize	Length of the *moduleName* buffer.
moduleName	Pointer to the buffer that is to receive the name of the Module.

Related Functions

TrkInitAssociationList
TrkGetNextAssociation

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationRevisionFixed

This function returns the revision number or version label representing when the defect reported by an issue was fixed.

C/C++

```
TRK_UINT TrkGetAssociationRevisionFixed(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR revisionFixed)/* Output */
```

Basic

```
Declare Function TrkGetAssociationRevisionFixed& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal bufferSize&,
    ByVal revisionFixed$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
bufferSize	Length of the *revisionFixed* buffer.
revisionFixed	Pointer to the buffer that is to receive the revision number or version label.

Related Functions

TrkSetAssociationRevisionFixed
TrkGetAssociationRevisionFound
TrkGetAssociationTimeFixed

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationRevisionFound

This function returns the revision number or version label representing when the defect reported by an issue was found.

C/C++

```
TRK_UINT TrkGetAssociationRevisionFound(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR revisionFound)/* Output */
```

Basic

```
Declare Function TrkGetAssociationRevisionFound& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal bufferSize&,
    ByVal revisionFound$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
bufferSize	Length of the *revisionFound* buffer.
revisionFound	Pointer to the buffer that is to receive the revision number or version label.

Related Functions

TrkSetAssociationRevisionFound
TrkGetAssociationRevisionFixed
TrkGetAssociationTimeFound

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationText

This function returns the change description of the current module association. You may call this function multiple times to read text data that exceeds the size of the *description* buffer.

C/C++

```
TRK_UINT TrkGetAssociationText(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT maxBufferSize,/* Input */
    TRK_STR description,  /* Output */
    TRK_UINT FAR *pDataRemaining)/* Output */
```

Basic

```
Declare Function TrkGetAssociationText& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal maxBufferSize&,
    ByVal description$, pDataRemaining&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
maxBufferSize	Length of the *description* buffer.
description	Pointer to the buffer that is to receive the description of the current module association.
pDataRemaining	Pointer to the buffer that is to receive the length of the remaining unread description data in the module association.

Related Functions

TrkSetAssociationText
TrkGetAssociationTextLength

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationTextLength

This function returns the length of the change description of the current module association.

C/C++

```
TRK_UINT TrkGetAssociationTextLength(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT FAR *pDataBufferSize)/* Output */
```

Basic

```
Declare Function TrkGetAssociationTextLength& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, pDataBufferSize&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
pDataBufferSize	Pointer to the buffer that is to receive the length of the description of the current module association.

Related Functions

TrkGetAssociationText
TrkSetAssociationText

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationTimeFixed

This function returns the timestamp representing when the defect reported by an issue was fixed.

C/C++

```
TRK_UINT TrkGetAssociationTimeFixed(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_TIME FAR *pTimeFixed)/* Output */
```

Basic

```
Declare Function TrkGetAssociationTimeFixed& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, pTimeFixed&)
```

Parameter

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
pTimeFixed	Pointer to the buffer that is to receive the timestamp.

Related Functions

TrkSetAssociationTimeFixed
TrkGetAssociationTimeFound
TrkGetAssociationRevisionFixed

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationTimeFound

This function returns the timestamp representing when the defect reported by an issue was found.

C/C++

```
TRK_UINT TrkGetAssociationTimeFound(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_TIME FAR *pTimeFound)/* Output */
```

Basic

```
TrkGetAssociationTimeFound& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, pTimeFound&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
pTimeFound	Pointer to the buffer that is to receive the timestamp.

Related Functions

TrkSetAssociationTimeFound
TrkGetAssociationRevisionFound
TrkGetAssociationTimeFixed

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAssociationTransactionID

This function returns the database event number for the transaction in which the current module association was created.

C/C++

```
TRK_UINT TrkGetAssociationTransactionID(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_TRANSACTION_ID FAR *pTransactionID)/* Output */
```

Basic

```
Declare Function TrkGetAssociationTransactionID& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, pTransactionID&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
pTransactionID	Pointer to the buffer that is to receive the Transaction ID.

Related Functions

TrkInitAssociationList
TrkGetNextAssociation

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations, Transaction IDs

TrkGetAssociationUser

This function returns the name of the user who created the current module association.

C/C++

```
TRK_UINT TrkGetAssociationUser(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR userName)     /* Output */
```

Basic

```
Declare Function TrkGetAssociationUser& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal bufferSize&, ByVal userName$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
bufferSize	Length of the *userName* buffer.
userName	Pointer to the buffer that is to receive the name of the user.

Related Functions

TrkSetAssociationUser
TrkGetAssociationText
TrkGetAssociationTextLength

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkGetAttachedFileName

This function returns the name of the current attached file.

C/C++

```
TRK_UINT TrkGetAttachedFileName(
    TRK_ATTFILE_HANDLE trkAttFileHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR filename)     /* Output */
```

Basic

```
Declare Function TrkGetAttachedFileName& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&, ByVal bufferSize&, ByVal filename$)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.
bufferSize	Length of *filename* buffer.
filename	Pointer to the buffer that is to receive the name of the attached file.

Related Functions

TrkGetAttachedFileTime
TrkExtractAttachedFile
TrkGetAttachedFileTransactionID

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, viewing
Tracker Toolkit Reference Guide	Attached Files

TrkGetAttachedFileStorageMode

This function returns a long integer value that represents the storage mode of the attached file. Possible values are `TRK_FILE_BINARY` and `TRK_FILE_ASCII`.

C/C++

```
TRK_UINT TrkGetAttachedFileStorageMode(
    TRK_ATTFILE_HANDLE trkAttFileHandle,/* Input */
    TRK_FILE_STORAGE_MODE FAR *pStorageMode)/* Output */
```

Basic

```
Declare Function TrkGetAttachedFileStorageMode& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&, pStorageMode&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.
pStorageMode	Pointer to the buffer that is to receive the storage mode of the attached file.

Related Functions

TrkAddNewAttachedFile
TrkExtractAttachedFile

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, viewing
Tracker Toolkit Reference Guide	Attached Files

TrkGetAttachedFileTime

This function returns a timestamp representing the creation time of the current attached file.

C/C++

```
TRK_UINT TrkGetAttachedFileTime(
    TRK_ATTFILE_HANDLE trkAttFileHandle,/* Input */
    TRK_TIME FAR *pTimeStamp)/* Output */
```

Basic

```
Declare Function TrkGetAttachedFileTime& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&, pTimeStamp&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.
pTimeStamp	Pointer to the buffer that is to receive the timestamp.

Related Functions

TrkGetAttachedFileName
TrkExtractAttachedFile
TrkGetAttachedFileTransactionID

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, viewing
Tracker Toolkit Reference Guide	Attached Files

TrkGetAttachedFileTransactionID

This function returns the database event number for the transaction in which the specified attached file was attached to its record.

C/C++

```
TRK_UINT TrkGetAttachedFileTransactionID(
    TRK_ATTFILE_HANDLE trkAttFileHandle,/* Input */
    TRK_TRANSACTION_ID FAR *pTransactionID)/* Output */
```

Basic

```
Declare Function TrkGetAttachedFileTransactionID& Lib "TRKTOOLN"
(ByVal trkAttFileHandle&, pTransactionID&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.
pTransactionID	Pointer to the buffer that is to receive the Transaction ID.

Related Functions

TrkGetAttachedFileName
TrkExtractAttachedFile
TrkGetAttachedFileData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Related files, viewing
Tracker Toolkit Reference Guide	Attached Files, Transaction IDs

TrkGetDescriptionAccessRights

This function returns the access rights for the description of the current record. Possible access right values are `TRK_READ_ONLY` and `TRK_READ_WRITE`. Access rights for a description depend on the rights of the logged in user and the state of the record in the database. For example, a description may be modifiable by the caller in a newly submitted record, but not upon update.

C/C++

```
TRK_UINT TrkGetDescriptionAccessRights(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_ACCESS_MODE FAR *pAccessMode)/* Output */
```

Basic

```
Declare Function TrkGetDescriptionAccessRights& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pAccessMode&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pAccessMode	Pointer to the buffer that is to receive the type of access rights assigned to the specified description.

Related Functions **TrkSetDescriptionData**

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Field permissions
Tracker Toolkit Reference Guide	Descriptions

TrkGetDescriptionData

This function returns the contents of the description of the current record. You may call this function multiple times to read description data that exceeds the size of the *data* buffer.

C/C++

```
TRK_UINT TrkGetDescriptionData(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_UINT maxBufferSize,/* Input */
    TRK_STR data          /* Output */
    TRK_UINT FAR* pDataRemaining/* Output */
```

Basic

```
Declare Function TrkGetDescriptionData& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal maxBufferSize&, ByVal data$,
    pDataRemaining&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
maxBufferSize	Length of the *data* buffer.
data	Pointer to the buffer that is to receive the contents of the description.
pDataRemaining	Pointer to the buffer that is to receive the length of remaining unread data.

Related Functions

TrkGetDescriptionDataLength
TrkSetDescriptionData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Descriptions

TrkGetDescriptionDataLength

This function returns the length of the data contained in the description of the current record.

C/C++

```
TRK_UINT TrkGetDescriptionDataLength(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_UINT FAR *pDataBufferSize)/* Output */
```

Basic

```
Declare Function TrkGetDescriptionDataLength& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pDataBufferSize&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pDataBufferSize	Pointer to the buffer that is to receive the length of the data in the description.

Related Functions

TrkGetDescriptionData
TrkSetDescriptionData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Descriptions

TrkGetDescriptionTransactionID

This function returns the database event number for the transaction in which the description of the current record was most recently changed.

C/C++

```
TRK_UINT TrkGetDescriptionTransactionID(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_TRANSACTION_ID FAR* pTransactionID)/* Output */
```

Basic

```
Declare Function TrkGetDescriptionTransactionID& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pTransactionID	Pointer to the buffer that is to receive the Transaction ID.

Related Functions

TrkInitDescriptionList
TrkGetNextDescription

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Descriptions, Transaction IDs

TrkGetFieldAccessRights

This function returns access rights for the specified field in the current record. Possible access right values are `TRK_READ_ONLY` and `TRK_READ_WRITE`. Access rights for a field depend on the rights of the login user and the state of the record in the database. For example, a field may be modifiable by the caller in a newly submitted record, but not upon update.

C/C++

```
TRK_UINT TrkGetFieldAccessRights(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_ACCESS_MODE FAR *pAccessMode)/* Output */
```

Basic

```
Declare Function TrkGetFieldAccessRights& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal fieldName$, pAccessMode&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose access rights you want to read.
pAccessMode	Pointer to the buffer that is to receive the type of access rights assigned to the specified field.

Related Functions

TrkNewRecordBegin
TrkNewRecordCommit
TrkUpdateRecordBegin
TrkUpdateRecordCommit

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Field permissions
Tracker Toolkit Reference Guide	Fields

TrkGetFieldDefaultNumericValue

This function returns the default numeric value of a field in the specified type of record.

C/C++

```
TRK_UINT TrkGetFieldDefaultNumericValue(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_UINT FAR *pDefaultValue)/* Output */
```

Basic

```
Declare Function TrkGetFieldDefaultNumericValue& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal fieldName$, ByVal recordType&,
    pDefaultValue&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose default value you want to read.
recordType	Integer representing the record type containing the specified field.
pDefaultValue	Pointer to the buffer that is to receive the default value of the field.

Related Functions

TrkGetFieldDefaultStringValue
TrkGetFieldStringValue
TrkGetNumericFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Custom Fields
Tracker Toolkit Reference Guide	Fields

TrkGetFieldDefaultStringValue

This function returns the default string value of a field in the specified type of record.

C/C++

```
TRK_UINT TrkGetFieldDefaultStringValue(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR defaultValue) /* Output */
```

Basic

```
Declare Function TrkGetFieldDefaultStringValue& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal fieldName$, ByVal recordType&,
    ByVal bufferSize&, ByVal defaultValue$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose default value you want to read.
recordType	Integer representing the record type containing the specified field.
bufferSize	Length of the *defaultValue* buffer.
defaultValue	Pointer to the buffer that is to receive the default value of the field.

Related Functions

TrkGetFieldDefaultNumericValue
TrkGetFieldStringValue
TrkGetStringFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Custom Fields
Tracker Toolkit Reference Guide	Fields

TrkGetFieldMaxDataLength

This function returns the maximum allowed data length of the specified field.

C/C++

```
TRK_UINT TrkGetFieldMaxDataLength(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_UINT FAR *pMaxDataLength)/* Output */
```

Basic

```
Declare Function TrkGetFieldMaxDataLength& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal fieldName$, ByVal recordType&,
    pMaxDataLength&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose maximum data length you want to read.
recordType	Integer that represents the type of record in which the field is located.
pMaxDataLength	Pointer to the buffer that is to receive the maximum allowed data length of the specified field.

Related Functions

TrkInitFieldList
TrkGetNextField
TrkGetFieldType

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Custom Fields
Tracker Toolkit Reference Guide	Fields

TrkGetFieldRange

This function returns the minimum and maximum possible values for the specified numeric field.

C/C++

```
TRK_UINT TrkGetFieldRange(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_UINT FAR *pMinValue,/* Output */
    TRK_UINT FAR *pMaxValue)/* Output */
```

Basic

```
Declare Function TrkGetFieldRange& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal fieldName$, ByVal recordType&,
    pMinValue&, pMaxValue&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose minimum and maximum values you want to read.
recordType	Integer representing the record type associated with the specified field.
pMinValue	Pointer to the buffer that is to receive the minimum value for the field.
pMaxValue	Pointer to the buffer that is to receive the maximum value for the field.

Related Functions

TrkGetFieldDefaultNumericValue
TrkGetFieldDefaultStringValue
TrkGetNumericFieldValue
TrkGetStringFieldValue

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Custom Fields
Tracker Toolkit Reference Guide	Fields

TrkGetFieldTransactionID

This function returns the database event number for the transaction in which the most recent change was made to the specified field in the current record.

C/C++

```
TRK_UINT TrkGetFieldTransactionID(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_TRANSACTION_ID FAR *pTransactionID)/* Output */
```

Basic

```
Declare Function TrkGetFieldTransactionID& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal fieldName$, pTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose Transaction ID you want to read.
pTransactionID	Pointer to the buffer that is to receive the Transaction ID of the specified field.

Related Functions

TrkGetNumericFieldValue
TrkGetStringFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	About Fields
Tracker Toolkit Reference Guide	Fields, Transaction IDs

TrkGetFieldType

This function returns a long integer value that represents the field type of the specified field.

C/C++

```
TRK_UINT TrkGetFieldType(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_FIELD_TYPE FAR *pFieldType)/* Output */
```

Basic

```
Declare Function TrkGetFieldType& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal fieldName$, ByVal recordType&,
    pFieldType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose type you wish to read.
recordType	Integer that represents the type of record in which the field is located.
pFieldType	Pointer to the buffer that is to receive the integer representing the field type.

Field Types

The following list shows the possible returned field types:

```
TRK_FIELD_TYPE_CHOICE
TRK_FIELD_TYPE_STRING
TRK_FIELD_TYPE_NUMBER
TRK_FIELD_TYPE_DATE
TRK_FIELD_TYPE_SUBMITTER
TRK_FIELD_TYPE_OWNER
TRK_FIELD_TYPE_USER
TRK_FIELD_TYPE_ELAPSED_TIME
```

Related Functions

TrkInitFieldList
TrkGetNextField
TrkGetFieldMaxDataLength

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	About Fields
Tracker Toolkit Reference Guide	Fields

TrkGetIniFile

This function returns the name and the path of the .INI file currently used by Tracker to provide default login information.

You may call this function before project login.

C/C++

```
TRK_UINT TrkGetIniFile(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR filename)     /* Output */
```

Basic

```
Declare Function TrkGetIniFile& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal fileName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *filename* buffer.
filename	Pointer to the buffer that is to receive the name and path of the .INI file currently in use.

Related Functions

TrkSetIniFile
TrkProjectLogin

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login

TrkGetLoginDBMSName

This function returns the name and path of the DBMS in use by the project that the user is currently logged in to.

C/C++

```
TRK_UINT TrkGetLoginDBMSName(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR DBMSName)     /* Output */
```

Basic

```
Declare Function TrkGetLoginDBMSName& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal DBMSName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *DBMSName* buffer.
DMBSName	Pointer to the buffer that is to receive the name and path of the DBMS.

Related Functions

TrkGetLoginUserName
TrkGetLoginProjectName
TrkGetLoginDBMSType
TrkProjectLogin

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Server
Tracker Toolkit Reference Guide	Project Login

TrkGetLoginDBMSType

This function returns the name of the DBMS for the project that the user is currently logged in to. The DBMSs supported by Tracker are Microsoft SQL Server, MSDE, Sybase, and Oracle.

C/C++

```
TRK_UINT TrkGetLoginDBMSType(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR DBMSType)     /* Output */
```

Basic

```
Declare Function TrkGetLoginDBMSType& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal DBMSType$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *DBMSType* buffer.
DBMSType	Pointer to the buffer that is to receive the name of the DBMS in use by the project.

Related Functions

TrkGetLoginUserName
TrkGetLoginProjectName
TrkGetLoginDBMSName
TrkProjectLogin

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Databases,supported
Tracker Toolkit Reference Guide	Project Login

TrkGetLoginProjectName

This function returns the name of the project that the user is currently logged in to.

C/C++

```
TRK_UINT TrkGetLoginProjectName(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR projectName)  /* Output */
```

Basic

```
Declare Function TrkGetLoginProjectName& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal projectName$)
```

Parameters

trkHandle	Handle allocated by **TrkHandleAlloc**.
bufferSize	Length of the *projectName* buffer.
projectName	Pointer to the buffer that is to receive the name of the project to which the caller is currently logged in.

Related Functions

TrkGetLoginUserName
TrkGetLoginDBMSType
TrkGetLoginDBMSName
TrkProjectLogin

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Projects
Tracker Toolkit Reference Guide	Project Login

TrkGetLoginUserName

This function returns the user name under which the user is currently logged in to a project.

C/C++

```
TRK_UINT TrkGetLoginUserName(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR userName)     /* Output */
```

Basic

```
Declare Function TrkGetLoginUserName& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal userName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *userName* buffer.
userName	Pointer to the buffer that is to receive the name of the user.

Related Functions

TrkGetLoginDBMSType
TrkGetLoginProjectName
TrkGetLoginDBMSName

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Administrator's Guide	Users
Tracker Toolkit Reference Guide	Project Login

TrkGetNextAssociation

This function provides sequential access to the module associations in an in-memory module association list.

C/C++

```
TRK_UINT TrkGetNextAssociation(
    TRK_ASSOC_HANDLE trkAssociationHandle)/* Input */
```

Basic

```
Declare Function TrkGetNextAssociation& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&)
```

Parameters

TrkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.

Related Functions

TrkInitAssociationList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Module Associations, Lists and Sequential Access

TrkGetNextAttachedFile

This function provides sequential access to the attached files in an in-memory attached file list.

C/C++

```
TRK_UINT TrkGetNextAttachedFile(
    TRK_ATTFILE_HANDLE trkAttFileHandle)/* Input */
```

Basic

```
Declare Function TrkGetNextAttachedFile& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.

Related Functions

TrkInitAttachedFileList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Attached Files, Lists and Sequential Access

TrkGetNextChoice

This function provides sequential access to the Choices in an in-memory Choice list.

C/C++

```
TRK_UINT TrkGetNextChoice(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR choiceName)   /* Output */
```

Basic

```
Declare Function TrkGetNextChoice& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal choiceName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *choiceName* buffer.
choiceName	Pointer to the buffer that is to receive the name of the current choice.

Related Functions

TrkInitChoiceList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Fields, Lists and Sequential Access

TrkGetNextDBMSType

This function provides sequential access to the DBMS in an in-memory DBMS list.

You may call this function before project login.

C/C++

```
TRK_UINT TrkGetNextDBMSType(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR DBMSType)     /* Output */
```

Basic

```
Declare Function TrkGetNextDBMSType& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal DBMSType$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *DBMSType* buffer.
DBMSType	Pointer to the buffer that is to receive the name of the current DBMS.

Related Functions

TrkInitDBMSTypeList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkGetNextField

This function provides sequential access to the fields in an in-memory field list.

C/C++

```
TRK_UINT TrkGetNextField(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR fieldName,    /* Output */
    TRK_FIELD_TYPE FAR *pFieldType)/* Output (optional) */
```

Basic

```
Declare Function TrkGetNextField& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal fieldName$,
    pFieldType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *fieldName* buffer.
fieldName	Pointer to the buffer that is to receive the name of the current field.
pFieldType	Pointer to the buffer that is to receive the integer representing the type of the current field.

Related Functions **TrkInitFieldList**

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Fields, Lists and Sequential Access

TrkGetNextIssueType

This function provides sequential access to the issue types in an in-memory issue-type list.

C/C++

```
TRK_UINT TrkGetNextIssueType(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR issueTypeName /* Output */
```

Basic

```
Declare Function TrkGetNextIssueType& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal userName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *issueTypeName* buffer.
issueTypeName	Pointer to the buffer that is to receive the name of the issue type.

Related Functions **TrkInitIssueTypeList**

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkGetNextNote

This function provides sequential access to the notes in an in-memory note list.

C/C++

```
TRK_UINT TrkGetNextNote(
    TRK_NOTE_HANDLE trkNoteHandle)/* Input */
```

Basic

```
Declare Function TrkGetNextNote& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&)
```

Parameters

trkNoteHandle Handle returned by **TrkNoteHandleAlloc**.

Related Functions

TrkNoteHandleAlloc
TrkInitNoteList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Notes, Lists and Sequential Access

TrkGetNextProject

This function provides sequential access to the projects in an in-memory project list.

C/C++

```
TRK_UINT TrkGetNextProject(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR projectName)  /* Output */
```

Basic

```
Declare Function TrkGetNextProject& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal projectName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *projectName* buffer.
projectName	Pointer to the buffer that is to receive the name of the current project.

Related Functions

TrkInitProjectList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkGetNextQueryName

This function provides sequential access to the Queries in an in-memory Query list.

C/C++

```
TRK_UINT TrkGetNextQueryName(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR queryName,    /* Output */
    TRK_RECORD_TYPE FAR *pRecordType)/* Output (optional) */
```

Basic

```
Declare Function TrkGetNextQueryName& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal queryName$,
    pRecordType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *queryName* buffer.
queryName	Pointer to the buffer that is to receive the name of the current Query.
pRecordType	Pointer to the buffer that is to receive the integer that represents the type of record for which the Query is defined.

Related Functions

TrkInitQueryNameList
TrkGetQueryRecordType

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Queries, saved
Tracker Toolkit Reference Guide	Queries, Lists and Sequential Access

TrkGetNextRecord

This function provides sequential access to the records in an in-memory record list.

C/C++

```
TRK_UINT TrkGetNextRecord(
    TRK_RECORD_HANDLE trkRecordHandle)/* Input */
```

Basic

```
Declare Function TrkGetNextRecord& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&)
```

Parameters

trkRecordHandle Handle returned by **TrkRecordHandleAlloc**.

Related Functions

TrkQueryInitRecordList
TrkInTrayInitRecordList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions, Lists and Sequential Access

TrkGetNextRecordType

This function provides sequential access to the record types in an in-memory record type list.

C/C++

```
TRK_UINT TrkGetNextRecordType(
    TRK_HANDLE trkHandle, /* Input */
    TRK_RECORD_TYPE FAR *pRecordType)/* Output */
```

Basic

```
Declare Function TrkGetNextRecordType& Lib "TRKTOOLN"
    (ByVal trkHandle&, pRecordType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
pRecordType	Pointer to the buffer that is to receive the record type.

Related Functions

TrkInitRecordTypeList
TrkGetRecordTypeName

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Types, Lists and Sequential Access

TrkGetNextUser

This function provides sequential access to the users in an in-memory user list.

C/C++

```
TRK_UINT TrkGetNextUser(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR userName)     /* Output */
```

Basic

```
Declare Function TrkGetNextUser& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal bufferSize&, ByVal userName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
bufferSize	Length of the *userName* buffer.
userName	Pointer to the buffer that is to receive the name of the current user.

Related Functions

TrkInitUserList
TrkGetUserFullName
TrkGetUserEmail

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkGetNoteAuthor

This function returns the name of the author of the current note.

C/C++

```
TRK_UINT TrkGetNoteAuthor(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR authorName)   /* Output */
```

Basic

```
Declare Function TrkGetNoteAuthor& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal bufferSize&, ByVal authorName$)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
bufferSize	Size of the *authorName* buffer.
authorName	Pointer to the buffer that is to receive the name of the author.

Related Functions

TrkSetNoteAuthor
TrkGetNoteCreateTime
TrkGetNoteModifyTime
TrkGetNoteTransactionID
TrkGetNoteData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkGetNoteCreateTime

This function returns a timestamp representing the time the current note was created.

C/C++

```
TRK_UINT TrkGetNoteCreateTime(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_TIME FAR* pCreateTime)/* Output */
```

Basic

```
Declare Function TrkGetNoteCreateTime& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, pCreateTime&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
pCreateTime	Pointer to the buffer that is to receive the timestamp representing the time the current note was created.

Related Functions

TrkSetNoteCreateTime
TrkGetNoteModifyTime
TrkGetNoteTransactionID

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkGetNoteData

This function retrieves the contents of the current note. You may call this function multiple times to read note data that exceeds the size of the *noteData* buffer.

C/C++

```
TRK_UINT TrkGetNoteData(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_UINT maxBufferSize,/* Input */
    TRK_STR noteData,     /* Output */
    TRK_UINT FAR *pDataRemaining)/* Output */
```

Basic

```
Declare Function TrkGetNoteData& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal maxBufferSize&, ByVal noteDate$,
    pDataRemaining&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
maxBufferSize	Length of the *noteData* buffer.
noteData	Pointer to the buffer that is to receive the contents of the current note.
pDataRemaining	Pointer to the buffer that is to receive the length of the remaining unread data in the note.

Related Functions

TrkGetNoteTitle
TrkGetNoteDataLength
TrkSetNoteTitle
TrkSetNoteData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkGetNoteDataLength

This function returns the length of the data contained in the current note.

C/C++

```
TRK_UINT TrkGetNoteDataLength(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_UINT FAR *pDataBufferSize)/* Output */
```

Basic

```
Declare Function TrkGetNoteDataLength& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, pDataBufferSize&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
pDataBufferSize	Pointer to the buffer that is to receive the length of the data in the note.

Related Functions

TrkGetNoteTitle
TrkGetNoteData
TrkSetNoteTitle
TrkSetNoteData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkGetNoteModifyTime

This function returns a timestamp representing the most recent time the current note was modified.

C/C++

```
TRK_UINT TrkGetNoteModifyTime(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_TIME FAR *pModifyTime)/* Output */
```

Basic

```
Declare Function TrkGetNoteModifyTime& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, pModifyTime&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
pModifyTime	Pointer to the buffer that is to receive the timestamp representing the most recent time the current note was modified.

Related Functions

TrkSetNoteModifyTime
TrkGetNoteCreateTime
TrkGetNoteTransactionID

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkGetNoteTitle

This function returns the title of the current note.

C/C++

```
TRK_UINT TrkGetNoteTitle(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR noteTitle)    /* Output */
```

Basic

```
Declare Function TrkGetNoteTitle& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal bufferSize&, ByVal noteTitle$)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
bufferSize	Length of the *noteTitle* buffer.
noteTitle	Pointer to the buffer that is to receive the title of the note.

Related Functions

TrkGetNoteDataLength
TrkGetNoteData
TrkSetNoteTitle
TrkSetNoteData

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkGetNoteTransactionID

This function returns the database event number for the transaction in which the current note was created or most recently modified.

C/C++

```
TRK_UINT TrkGetNoteTransactionID(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_TRANSACTION_ID FAR *pTransactionID)/* Output */
```

Basic

```
Declare Function TrkGetNoteTransactionID& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, pTransactionID&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
pTransactionID	Pointer to the buffer that is to receive the Transaction ID.

Related Functions

TrkGetNoteCreateTime
TrkGetNoteModifyTime
TrkSetNoteCreateTime
TrkSetNoteModifyTime

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes, Transaction IDs

TrkGetNumericAttribute

This function retrieves an (ID, value) pair managed by the Tracker Toolkit.

C/C++

```
TRK_UINT TrkGetNumericAttribute(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT attributeID, /* Input */
    TRK_UINT pValue)      /* Output */
```

Basic

```
Declare Function TrkGetNumericAttribute& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal attributeID&, pValue&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
attributeID	The ID component of the pair.
pValue	Pointer to the buffer that is to receive the value component of the pair.

Related Functions **TrkSetNumericAttribute**

TrkGetNumericFieldValue

This function returns the numeric value of the specified field in the current record.

C/C++

```
TRK_UINT TrkGetNumericFieldValue(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_UINT FAR *pFieldValue)/* Output */
```

Basic

```
Declare Function TrkGetNumericFieldValue& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal fieldName$, pFieldValue&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose value you want to read.
pFieldValue	Pointer to the buffer that is to receive the numeric value of the field.

Note that signed and unsigned ints are stored identically, so you need only cast the value to the appropriate type:

```
int value;
error = TrkGetNumericFieldValue(
    trkRecordHandle,
    fieldName
    (unsigned*) &value);
```

Related Functions

TrkGetStringFieldValue
TrkSetNumericFieldValue
TrkSetStringFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Fields

TrkGetProjectTransactionID

This function returns the last allocated Transaction ID for a record of the specified type in the project to which the user is currently logged in.

C/C++

```
TRK_UINT TrkGetProjectTransactionID(
    TRK_HANDLE trkHandle, /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_TRANSACTION_ID FAR *pTransactionID/* Output */
```

Basic

```
Declare Function TrkGetProjectTransactionID& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal recordType&, pTransactionID&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
recordType	Integer which represents the record type.

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Transaction IDs

TrkGetQueryRecordType

This function returns a long integer value representing the record type associated with the current Query.

C/C++

```
TRK_UINT TrkGetQueryRecordType(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR queryName,   /* Input */
    TRK_RECORD_TYPE FAR *pRecordType)/* Output */
```

Basic

```
Declare Function TrkGetQueryRecordType& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal queryName$, pRecordType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
queryName	Pointer to the name of the Query whose record type you want to read.
pRecordType	Pointer to the buffer that is to receive the record type.

Related Functions

TrkInitQueryNameList
TrkGetNextQueryName
TrkQueryInitRecordList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Record Types

TrkGetRecordRecordType

This function returns the record type of the current record.

C/C++

```
TRK_UINT TrkGetRecordRecordType(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_RECORD_TYPE FAR *pRecordType)/* Output */
```

Basic

```
Declare Function TrkGetRecordRecordType& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pRecordType&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pRecordType	Pointer to the buffer that is to receive the record type.

Related Functions

TrkGetRecordTypeName

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Record Types

TrkGetRecordTransactionID

This function returns the database event numbers for the transactions in which the current record was submitted and most recently updated.

C/C++

```
TRK_UINT TrkGetRecordTransactionID(
    TRK_RECORD_HANDLE trkRecordHandle,           /* Input */
    TRK_TRANSACTION_ID FAR *pSubmitTransactionID,/* Output
    (optional)*/
    TRK_TRANSACTION_ID FAR *pUpdateTransactionID)/* Output
    (optional)*/
```

Basic

```
Declare Function TrkGetRecordTransactionID& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pSubmitTransactionID&,
    pUpdateTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pSubmitTransactionID	Pointer to the buffer that is to receive the submit Transaction ID.
pUpdateTransactionID	Pointer to the buffer that is to receive the update Transaction ID.

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Record Transactions, Transaction IDs

TrkGetRecordTypeName

This function returns the name of a record type, given an integer value returned by **TrkGetNextRecordType**, **TrkGetNextQueryName**, **TrkGetRecordRecordType** or **TrkGetQueryRecordType**.

Possible record type names are “System Change Request” and “Time Record”.

C/C++

```
TRK_UINT TrkGetRecordTypeName
    TRK_HANDLE trkHandle, /* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR recordTypeName)/* Output */
```

Basic

```
Declare Function TrkGetRecordTypeName& Lib “TRKTOOLN”
    (ByVal trkHandle&, ByVal recordType&, ByVal bufferSize&,
    ByVal recordTypeName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
recordType	Integer value that represents the record type.
bufferSize	Length of the *recordTypeName* buffer.
recordTypeName	Pointer to the buffer that is to receive the name of the record type.

Related Functions

TrkInitRecordTypeList
TrkGetNextRecordType

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User’s Guide	Issues
Tracker Toolkit Reference Guide	Record Types

TrkGetSingleRecord

This function makes a single record current, given its record ID and record type. The record Id is automatically assigned by Tracker when a new record is submitted and is stored in the "Id" String Field of each record.

C/C++

```
TRK_UINT TrkGetSingleRecord(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_UINT recordId,    /* Input */
    TRK_RECORD_TYPE recordType)/* Input */
```

Basic

```
Declare Function TrkGetSingleRecord& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal recordId&, ByVal recordType&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
recordId	The external database number assigned to each record in the Tracker database.
recordType	Integer that represents the record type of the record you want to make current.

Related Functions

TrkQueryInitRecordList
TrkInTrayInitRecordList
TrkGetNextRecord

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Record Types, Lists and Sequential Access

TrkGetStringFieldValue

This function returns a string value from the specified field in the current record.

C/C++

```
TRK_UINT TrkGetStringFieldValue(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_UINT fieldValueBufferSize,/* Input */
    TRK_STR fieldValue)   /* Output */
```

Basic

```
Declare Function TrkGetStringFieldValue& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal fieldName$,
    ByVal fieldValueBufferSize&, ByVal fieldValue$)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldName	Pointer to a string containing the name of the field whose value you want to read.
fieldValueBufferSize	Length of the *fieldValue* buffer.
fieldValue	Pointer to the buffer that is to receive the value of the specified string or Choice field.

Related Functions

TrkGetNumericFieldValue
TrkSetNumericFieldValue
TrkSetStringFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Issues
Tracker Toolkit Reference Guide	Fields

TrkGetUserEmail

This function returns the e-mail address of the specified user, as defined in the Tracker Administrator application.

C/C++

```
TRK_UINT TrkGetUserEmail(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR userName,    /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR emailAddress) /* Output */
```

Basic

```
Declare Function TrkGetUserEmail& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal userName$, ByVal bufferSize&,
    ByVal emailAddress$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
userName	Pointer to a string containing the user name of the user whose e-mail address you want to read.
bufferSize	Length of *emailAddress* buffer.
emailAddress	Pointer to the buffer that is to receive the e-mail address of the current user.

Related Functions **TrkGetUserFullName**

Related Topics For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Users
Tracker Toolkit Reference Guide	Project Login

TrkGetUserFullName

This function returns the full name of the specified user, as defined in the Tracker Administrator application.

C/C++

```
TRK_UINT TrkGetUserFullName(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR userName,    /* Input */
    TRK_UINT bufferSize,  /* Input */
    TRK_STR fullName)     /* Output */
```

Basic

```
Declare Function TrkGetUserFullName& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal userName$, ByVal bufferSize&,
    ByVal fullName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
userName	Pointer to a string containing the user name of the user whose full name you want to read.
bufferSize	Length of the *fullName* buffer.
fullName	Pointer to the buffer that is to receive the full name of the specified user.

Related Functions **TrkGetUserEmail**

Related Topics For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Users
Tracker Toolkit Reference Guide	Project Login

TrkHandleAlloc

This function provides a Tracker Handle and allocates internal resources for use with all Tracker Toolkit functions. You must call **TrkHandleAlloc** before you call any other Tracker functions. The handle you obtain remains open until you call **TrkHandleFree**.

C/C++

```
TRK_UINT TrkHandleAlloc(
    TRK_VERSION_ID_TYPE trkVersionID,/* Input */
    TRK_HANDLE FAR *pTrkHandle)/* Output */
```

Basic

```
Declare Function TrkHandleAlloc& Lib "TRKTOOLN"
    (ByVal trkVersionID&, pTrkHandle&)
```

Parameters

trkVersionID	Integer value that represents the version of the Toolkit currently in use. Pass the defined constant `TRK_VERSION_ID` with this parameter.
pTrkHandle	Pointer to the variable that is to receive the Tracker Handle.

Special Considerations

If the Toolkit Version ID doesn't match the version of Tracker you have installed, this function will return a value of `TRK_E_VERSION_MISMATCH`.

Related Functions

TrkHandleFree

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles

TrkHandleFree

This function closes files and releases internal resources associated with a previously obtained Tracker Handle. Call this function when you have completed all operations related to the Tracker Handle.

C/C++

```
TRK_UINT TrkHandleFree(
    TRK_HANDLE FAR *pTrkHandle)/* Input/Output */
```

Basic

```
Declare Function TrkHandleFree& Lib "TRKTOOLN"
    (pTrkHandle&)
```

Parameters

pTrkHandle Handle returned by **TrkHandleAlloc**.

Related Functions

TrkHandleAlloc

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles

TrkImportHandleAlloc

This function allocates an Import Handle, which is a required parameter for all Tracker import functions. The handle you obtain remains open until you call **TrkImportHandleFree**.

C/C++

```
TRK_UINT TrkImportHandleAlloc(
    TRK_HANDLE trkHandle, /* Input */
    TRK_IMPORT_HANDLE FAR *pTrkImportHandle)/* Output */
```

Basic

```
Declare Function TrkImportHandleAlloc& Lib "EXPDLLW"
    (ByVal trkHandle&, pTrkImportHandle&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
pTrkImportHandle	Pointer to the variable that is to receive the Import Handle.

Related Functions

TrkImportHandleFree
TrkImportInit
TrkImportNewRecords

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Importing Records
Tracker Toolkit Reference Guide	Handles

TrkImportHandleFree

This function closes the specified Import Handle and releases all memory associated with it. Call this function when you have completed all operations related to the Import Handle.

C/C++

```
TRK_UINT TrkImportHandleFree(
    TRK_IMPORT_HANDLE FAR *pTrkImportHandle)/* Input/Output */
```

Basic

```
Declare Function TrkImportHandleFree& Lib "EXPDLLW"
    (pTrkImportHandle&)
```

Parameters

pTrkImportHandle Handle returned by **TrkImportHandleAlloc**.

Related Functions

TrkImportHandleAlloc
TrkImportInit
TrkImportNewRecords

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Importing Records
Tracker Toolkit Reference Guide	Handles

TrkImportInit

This function initializes the record import feature and allows you to specify the name of the file from which you want to import the information. To further set import options, call **TrkImportSetOptions** after you call this function. Then call **TrkImportNewRecords** to import the records from a text file.

C/C++

```
TRK_UINT TrkImportInit(
    TRK_IMPORT_HANDLE trkImportHandle,/* Input */
    TRK_CSTR fileToReadFrom,/* Input */
    TRK_BOOL trackerFormat,/* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_CHAR delimiterCharacter,/* Input (optional) */
    TRK_CHAR separatorCharacter,/* Input (optional) */
    TRK_DATE_FORMAT dateFormat,/* Input (optional) */
    TRK_TIME_FORMAT timeFormat)/* Input (optional) */
```

Basic

```
Declare Function TrkImportInit& Lib "EXPDLLW"
    (ByVal trkImportHandle&, ByVal fileToReadFrom$,
    ByVal trackerFormat&, ByVal delimiterCharacter&,
    ByVal separatorCharacter&, ByVal dateFormat&,
    ByVal timeFormat&)
```

Parameters

trkImportHandle	Handle returned by **TrkImportHandleAlloc**.
fileToReadFrom	Pointer to a string that contains the name of the file from which you want to import the data.
trackerFormat	Possible values are `TRUE` and `FALSE`. You must set this parameter to `TRUE` if the data you are importing was previously exported in Tracker format.
recordType	Integer value that represents the record type of the record you want to import. This parameter is not required if *trackerFormat* is `TRUE`.

delimiterCharacter ASCII value of the character you want to use to mark the beginning and end of each field in the text file. This parameter is not required if *trackerFormat* is TRUE.

separatorCharacter ASCII value of the character you want to use to separate fields in the text file. This parameter is not required if *trackerFormat* is TRUE.

dateFormat Integer value representing the type of imbedded quote you want to use in the resulting text file. Possible values are TRK_CONTROL_PANEL_DATE and TRK_DBASE_FORMAT. This parameter is not required if *trackerFormat* is TRUE.

The control panel date format is the default Windows date format. The dBase format is YYYYMMDD. This parameter is not required if *trackerFormat* is TRUE.

timeFormat Integer value representing the time format you want to use in the resulting text file. Possible values are TRK_CONTROL_PANEL_TIME (the default Windows format), TRK_24HOUR and TRK_24HOUR_LEADING_ZERO. This parameter is not required if *trackerFormat* is TRUE.

Related Functions

TrkImportSetOptions
TrkImportNewRecords
TrkExportRecordsBegin
TrkExportOneRecord
TrkExportRecordsClose

Related Topics For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Importing Records
Tracker Toolkit Reference Guide	Importing and Exporting

TrkImportNewRecords

This function imports record information from a text file. You must call **TrkImportInit** to initialize the import feature before you call this function, and call **TrkImportSetOptions** if you want to set import options.

C/C++

```
TRK_UINT TrkImportNewRecords(
    TRK_IMPORT_HANDLE trkImportHandle,/* Input */
    TRK_CSTR errorLogFile)/* Input (optional) */
```

Basic

```
Declare Function TrkImportNewRecords& Lib "EXPDLLW"
    (ByVal trkImportHandle&, ByVal errorLogFile$)
```

Parameters

trkImportHandle	Handle returned by **TrkImportHandleAlloc**.
errorLogFile	Pointer to a string that contains the name and path of the file to which you want to log import error information.

Related Functions

TrkImportInit
TrkImportSetOptions
TrkExportRecordsBegin
TrkExportOneRecord
TrkExportRecordsClose

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Importing Records
Tracker Toolkit Reference Guide	Importing Records

TrkImportSetOptions

Call this function after you call **TrkImportInit** to set the options for the import feature. If you do not call this function, the default values will be assumed. Once you have set import options, call **TrkImportNewRecords** to import records from the text file.

If you set an option to fail, the entire import process will fail if a bad value is encountered.

C/C++

```
TRK_UINT TrkImportSetOptions(
    TRK_IMPORT_HANDLE trkImportHandle,/* Input */
    TRK_CHOICE_OPTION choiceOption,/* Input (optional) */
    TRK_USER_OPTION userOption,/* Input (optional) */
    TRK_NUMBER_OPTION numberOption,/* Input (optional) */
    TRK_DATE_OPTION dateOption,/* Input (optional) */
    TRK_CSTR defaultDate) /* Input (optional) */
```

Basic

```
Declare Function TrkImportSetOptions& Lib "EXPDLLW"
    (ByVal trkImportHandle&, ByVal choiceOption&, ByVal userOption&,
    ByVal numberOption&, ByVal dateOption&, ByVal defaultDate$)
```

Parameters

trkImportHandle Handle returned by **TrkImportHandleAlloc**.

choiceOption Integer indicating the action you wish the Tracker Toolkit to take if a bad Choice value is encountered in the text file.

Possible values are `TRK_FAIL_CHOICE` (the default), `TRK_DEFAULT_CHOICE`, and `TRK_NEW_CHOICE`. Use `TRK_FAIL_CHOICE` to indicate that you want the import to fail should a bad Choice value be encountered. Use `TRK_DEFAULT_CHOICE` to indicate that you want the default field value to be substituted. Use `TRK_NEW_CHOICE` to indicate that you want a new choice to be added should the Toolkit encounter a choice in the import file that does not exist in the database.

userOption

Integer indicating the action you wish the Tracker Toolkit to take if a missing or unidentified user name is encountered in the text file.

Possible values are `TRK_FAIL_USER`, which is the default, and `TRK_ADD_USER`. Use `TRK_FAIL_USER` to indicate that you want the import to fail should a bad user be encountered. Use `TRK_ADD_USER` to add the user name to the project.

numberOption

Integer that indicates the action you want the Tracker Toolkit to take should it encounter an out-of-range number in the text file.

Possible values are `TRK_FAIL_NUMBER`, which is the default, and `TRK_DEFAULT_NUMBER`. Use `TRK_FAIL_NUMBER` to indicate that you want import to fail should a bad number be encountered. Use `TRK_DEFAULT_NUMBER` to indicate that you want to substitute the default value for the field.

dateOption

Integer that indicates the action you want the Tracker Toolkit to take should it encounter a bad date in the text file.

Possible values are `TRK_FAIL_DATE`, which is the default, `TRK_SET_CURRENT` and `TRK_SET_TO_SPECIFIED`. Use `TRK_FAIL_DATE` to indicate that you want the import to fail should a bad date be encountered. Use `TRK_SET_CURRENT` to indicate that you want the Toolkit to substitute the current date. Use

	`TRK_SET_TO_SPECIFIED` to indicate that you want to substitute the value passed in the *`defaultDate`* parameter.
`defaultDate`	Pointer to a string containing the date you want substituted for a bad date in the text file.

Related Functions

TrkImportInit
TrkImportNewRecords
TrkExportRecordsBegin
TrkExportOneRecord
TrkExportRecordsClose

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Importing Records
Tracker Toolkit Reference Guide	Importing Records

TrkInitAssociationList

This function initializes an in-memory list of the module associations attached to the current record.

C/C++

```
TRK_UINT TrkInitAssociationList(
    TRK_ASSOC_HANDLE trkAssociationHandle)/* Input */
```

Basic

```
Declare Function TrkInitAssociationList& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.

Related Functions

TrkAssociationHandleAlloc
TrkGetNextAssociation

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Module Associations, Lists and Sequential Access

TrkInitAttachedFileList

This function initializes an in-memory list of the attached files associated with the current record.

C/C++

```
TRK_UINT TrkInitAttachedFileList(
    TRK_ATTFILE_HANDLE trkAttFileHandle)/* Input */
```

Basic

```
Declare Function TrkInitAttachedFileList& Lib "TRKTOOLN"
    (ByVal trkAttFileHandle&)
```

Parameters

trkAttFileHandle	Handle returned by **TrkAttachedFileHandleAlloc**.

Related Functions

TrkAttachedFileHandleAlloc
TrkGetNextAttachedFile

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Importing Records
Tracker Toolkit Reference Guide	Attached Files, Lists and Sequential Access

TrkInitChoiceList

This function initializes an in-memory list of the Choices associated with the specified Choice field.

C/C++

```
TRK_UINT TrkInitChoiceList(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_RECORD_TYPE recordType)/* Input */
```

Basic

```
Declare Function TrkInitChoiceList& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal fieldName$, ByVal recordType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
fieldName	Pointer to a string containing the name of the Choice field whose Choices you want to list.
recordType	Integer that represents the record type whose Choices you want to list.

Related Functions

TrkGetNextChoice

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Fields, Lists and Sequential Access

TrkInitDBMSTypeList

This function initializes an in-memory list of all DBMS's known to Tracker.

You may call this function before project login.

C/C++

```
TRK_UINT TrkInitDBMSTypeList(
    TRK_HANDLE trkHandle) /* Input */
```

Basic

```
Declare Function TrkInitDBMSTypeList& Lib "TRKTOOLN"
    (ByVal trkHandle&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.

Related Functions

TrkGetNextDBMSType

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkInitFieldList

This function initializes an in-memory list of the fields associated with the specified record type.

C/C++

```
TRK_UINT TrkInitFieldList(
    TRK_HANDLE trkHandle, /* Input */
    TRK_RECORD_TYPE recordType)/* Input */
```

Basic

```
Declare Function TrkInitFieldList& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal recordType&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
recordType	Integer that represents the record type whose associated fields you want to list.

Related Functions

TrkGetNextField

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Fields, Lists and Sequential Access

TrkInitIssueTypeList

This function initializes an in-memory list of the issue types defined in the current project.

C/C++

```
TRK_UINT TrkInitIssueTypeList(
    TRK_HANDLE trkHandle  /* Input */
```

Basic

```
Declare Function TrkInitIssueTypeList& Lib "TRKTOOLN"
    (ByVal trkHandle&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.

Related Functions

TrkGetNextIssueType

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkInitNoteList

This function initializes an in-memory list of the notes attached to the current record.

C/C++

```
TRK_UINT TrkInitNoteList(
    TRK_NOTE_HANDLE trkNoteHandle)/* Input */
```

Basic

```
Declare Function TrkInitNoteList& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&)
```

Parameters

trkNoteHandle Handle returned by **TrkNoteHandleAlloc**.

Related Functions

TrkNoteHandleAlloc
TrkGetNextNote

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Notes, Lists and Sequential Access

TrkInitProjectList

This function initializes an in-memory list of the projects associated with the specified DBMS Type and DBMS Name.

Use the *DBMSLoginMode* parameter to specify the source of login information used for this function. Possible sources are the currently used .INI file, the default login information defined by a Tracker Administrator, or the information passed by the function's parameters.

If the *trkHandle* is logged in, all other parameters are ignored and the server associated with *trkHandle* is used.

C/C++

```
TRK_UINT TrkInitProjectList(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR DBMSType,    /* Input (optional) */
    TRK_CSTR DBMSName,    /* Input (optional) */
    TRK_CSTR DBMSUserName,/* Input (optional) */
    TRK_CSTR DBMSPassword,/* Input (optional) */
    TRK_DBMS_LOGIN_MODE DBMSLoginMode)/* Input */
```

Basic

```
Declare Function TrkInitProjectList& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal DMBSType$, ByVal DBMSName$,
    ByVal DBMSUserName$, ByVal DBMSPassword$, ByVal DBMSLoginMode&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
DBMSType	Pointer to a string containing the name of an DBMS type; for example, "Oracle".
DBMSName	Pointer to a string that contains the name and path of a DBMS; for example, "m:\merant\tracker\db".
DBMSUserName	Pointer to a string containing a DBMS user name.
DBMSPassword	Pointer to a string containing a DBMS password.

DBMSLoginMode	Integer value that specifies the login option for the function. Possible values are `TRK_USE_INI_FILE_DBMS_LOGIN`, `TRK_USE_SPECIFIED_DBMS_LOGIN` and `TRK_USE_DEFAULT_DBMS_LOGIN`. If you pass `TRK_USE_SPECIFIED_DBMS_LOGIN` for this parameter, the optional parameters are used; otherwise, they are ignored.

Related Functions

TrkGetNextProject
TrkProjectLogin

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkInitProjectListEx

This function initializes an in-memory list of the projects associated with a server name.

Use the *serverName* parameter to specify the source of login information used for this function. (Server names are created by the Tracker Server Wizard, which can be reached from the Open Project dialog box.)

If the *trkHandle* is logged in, the *serverName* is ignored and the server associated with *trkHandle* is used.

C/C++

```
TRK_UINT TrkInitProjectListEx(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR serverName)  /* Input */
```

Basic

```
Declare Function TrkInitProjectListEx& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal serverName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
serverName	Pointer to a string containing a server name; for example, "QA_server".

Related Functions

TrkGetNextProject
TrkInitProjectList
TrkProjectLoginEx

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkInitQueryNameList

This function initializes an in-memory list of the stored Queries defined in the current project.

C/C++

```
TRK_UINT TrkInitQueryNameList(
    TRK_HANDLE trkHandle) /* Input */
```

Basic

```
Declare Function TrkInitQueryNameList& Lib "TRKTOOLN"
    (ByVal trkHandle&)
```

Parameters

trkHandle Handle returned by **TrkHandleAlloc**.

Related Functions

TrkGetNextQueryName

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Lists and Sequential Access

TrkInitRecordTypeList

This function initializes an in-memory list of the record types defined for the current project.

C/C++

```
TRK_UINT TrkInitRecordTypeList(
    TRK_HANDLE trkHandle) /* Input */
```

Basic

```
Declare Function TrkInitRecordTypeList& Lib "TRKTOOLN"
    (ByVal trkHandle&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.

Related Functions

TrkGetNextRecordType
TrkGetRecordTypeName

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Types, Lists and Sequential Access

TrkInitUserList

This function initializes an in-memory list of the users defined in the current project.

C/C++

```
TRK_UINT TrkInitUserList(
    TRK_HANDLE trkHandle) /* Input */
```

Basic

```
Declare Function TrkInitUserList& Lib "TRKTOOLN"
    (ByVal trkHandle&)
```

Parameters

trkHandle Handle returned by **TrkHandleAlloc**.

Related Functions

TrkGetNextUser

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login, Lists and Sequential Access

TrkInTrayInitRecordList

This function initializes an in-memory list of records residing in the In Tray of the logged in user. You must specified the record type of the records you want to list.

The *transactionID* parameter allows you to specify a starting database event number for the records in the list. For example, if you pass the integer value 36 to the *transactionID* parameter, only records that have been modified in database transactions with event numbers later than 36 will be listed.

Optionally, this function returns a Transaction ID for the most recent transaction in the database by way of the *pNewTransactionID* parameter.

C/C++

```
TRK_UINT TrkInTrayInitRecordList(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_RECORD_TYPE recordType,/* Input */
    TRK_TRANSACTION_ID transactionID,/* Input (optional) */
    TRK_TRANSACTION_ID FAR *pNewTransactionID)/* Output (optional) */
```

Basic

```
Declare Function TrkInTrayInitRecordList& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal recordType&, ByVal transactionID&,
    pNewTransationID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
recordType	Integer value representing the record type of the records you want to list.
transactionID	The starting database event number for the list.
pNewTransactionID	Pointer to the buffer that is to receive the event number of the most recent transaction in the database.

Related Functions

TrkRecordHandleAlloc
TrkQueryInitRecordList
TrkGetNextRecord

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Lists and Sequential Access

TrkModifyIssueType

This function sets the issue type in the current record. You call this function within the **TrkUpdateRecordBegin/ TrkUpdateRecordCommit** function block or in a **TrkNewRecordBegin/TrkNewRecordCommit** function block. If you do not call this function when you submit a new record, the issue type for the new rcord is the default issue type.

C/C++

```
TRK_UINT TrkModifyIssueType(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldValue)  /* Input */
```

Basic

```
Declare Function TrkModifyIssueType& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal fieldValue$)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldValue	Pointer to the string value to which you want to set the issue type.

Related Functions

TrkGetNumericFieldValue
TrkGetStringFieldValue
TrkSetNumericFieldValue
TrkSetStringFieldValue

TrkNewRecordBegin

This function clears the current in-memory record and sets its fields to their default values. Call this function to begin a record transaction that submits a new record to the database. When you are finished setting field values or other data associated with the record, call **TrkNewRecordCommit**.

C/C++

```
TRK_UINT TrkNewRecordBegin(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_RECORD_TYPE recordType)/* Input */
```

Basic

```
Declare Function TrkNewRecordBegin& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal recordType&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
recordType	Integer value that represents the type of record you want to submit.

Related Functions

TrkNewRecordCommit
TrkRecordCancelTransaction

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions, Record Types

TrkNewRecordBeginEx

This function clears the current in-memory record and sets its fields to their default values. Call this function to begin a record transaction that submits a new record of the specified issue type to the database. When you are finished setting field values or other data associated with the record, call **TrkNewRecordCommit**.

C/C++

```
TRK_UINT TrkNewRecordBeginEx(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR issueTypeName)/* Input */
```

Basic

```
Declare Function TrkNewRecordBeginEx& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal issueTypeName$)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
issueTypeName	Pointer to the string value for the issue type to submit.

Related Functions **TrkNewRecordBegin**

TrkNewRecordCommit

This function submits a new record to the database and ends the record transaction. It also returns a Transaction ID representing the database event number for the submit transaction. You must call **TrkNewRecordBegin** before you can use this function.

C/C++

```
TRK_UINT TrkNewRecordCommit(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_TRANSACTION_ID FAR *pNewTransactionID)/* Output */
```

Basic

```
Declare Function TrkNewRecordCommit& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pNewTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pNewTransactionID	Pointer to the buffer that is to receive the new Transaction ID.

Related Functions

TrkNewRecordBegin
TrkRecordCancelTransaction

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions, Transaction IDs

TrkNoteHandleAlloc

This function allocates a Note Handle, which is a required parameter for all Tracker functions that operate on note data. The handle you obtain remains open until you call **TrkNoteHandleFree**.

C/C++

```
TRK_UINT TrkNoteHandleAlloc(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_NOTE_HANDLE FAR *pTrkNoteHandle)/* Output */
```

Basic

```
Declare Function TrkNoteHandleAlloc& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pTrkNoteHandle&)
```

Parameters

trkRecordHandle	Handle allocated by **TrkRecordHandleAlloc**.
pTrkNoteHandle	Pointer to the variable that is to receive the Note Handle.

Related Functions

TrkNoteHandleFree

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles, Notes

TrkNoteHandleFree

This function closes the specified Note Handle and releases all memory associated with it. Call this function when you have completed all operations related to the Note Handle.

C/C++

```
TRK_UINT TrkNoteHandleFree(
    TRK_NOTE_HANDLE FAR *pTrkNoteHandle)/* Input/Output */
```

Basic

```
Declare Function TrkNoteHandleFree& Lib "TRKTOOLN"
    (pTrkNoteHandle&)
```

Parameters

pTrkNoteHandle Handle returned by **TrkNoteHandleAlloc**.

Related Functions

TrkNoteHandleAlloc

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles, Notes

TrkProjectLogin

This function logs in to a Tracker project, using the specified login parameters.

Use the *DBMSLoginMode* parameter to specify the source of login information used for this function. Possible sources are the currently used .INI file, the default login information defined by a Tracker Administrator, or the information passed by the function's parameters.

C/C++

```
TRK_UINT TrkProjectLogin(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR userName,    /* Input (optional) */
    TRK_CSTR password,    /* Input (optional) */
    TRK_CSTR projectName, /* Input (optional) */
    TRK_CSTR DBMSType,    /* Input (optional) */
    TRK_CSTR DBMSName,    /* Input (optional) */
    TRK_CSTR DBMSUserName,/* Input (optional) */
    TRK_CSTR DBMSPassword,/* Input (optional) */
    TRK_DBMS_LOGIN_MODE DBMSLoginMode)/* Input */
```

Basic

```
Declare Function TrkProjectLogin& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal userName$, ByVal password$,
    ByVal projectName$, ByVal DBMSType$, ByVal DBMSName$,
    ByVal DBMSUserName$, ByVal DBMSPassword$, ByVal DBMSLoginMode&)
```

Parameters

trkHandle — Handle returned by **TrkHandleAlloc**.

userName — Pointer to a string that contains the user name which you want to use to log in to the project.

password — Pointer to a string that contains the Tracker password associated with the specified user login.

projectName — Pointer to a string that contains the name of the project to which you want to log in.

DBMSType — Pointer to a string that contains the name of the DBMS type used by the project you want to log in to; for example, "Oracle".

DBMSName	Pointer to a string that contains the name and path of the DBMS used by the project you want to log in to; for example, "m:\merant\tracker\db".
DBMSUserName	Pointer to a string that contains the DBMS user name associated with the specified user.
DBMSPassword	Pointer to a string that contains the DBMS password associated with the specified user login.
DBMSLoginMode	Integer value that specifies the login option for the function. Possible values are `TRK_USE_INI_FILE_DBMS_LOGIN`, `TRK_USE_SPECIFIED_DBMS_LOGIN` and `TRK_USE_DEFAULT_DBMS_LOGIN`. If you pass `TRK_USE_SPECIFIED_DBMS_LOGIN` for this parameter, the optional parameters are used; otherwise, they are ignored.

Related Functions

TrkProjectLogout
TrkGetIniFile
TrkSetIniFile

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login
Tracker Administrator's Guide	Databases, supported

TrkProjectLoginEx

This function logs in to a Tracker project, using the specified login parameters.

Use the *serverName* parameter to specify the source of login information used for this function. (Server names are created by the Tracker Server Wizard, which can be reached from the Open Project dialog box.)

Note that, unlike TrkProjectLogin(), each parameter is required for this function.

C/C++

```
TRK_UINT TrkProjectLoginEx(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR userName,    /* Input */
    TRK_CSTR password,    /* Input */
    TRK_CSTR projectName, /* Input */
    TRK_CSTR serverName)  /* Input */
```

Basic

```
Declare Function TrkProjectLoginEx& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal userName$, ByVal password$,
    ByVal projectName$, ByVal serverName$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
userName	Pointer to a string that contains the user name which you want to use to log in to the project.
password	Pointer to a string that contains the Tracker password associated with the specified user login.
projectName	Pointer to a string that contains the name of the project to which you want to log in.
serverName	Pointer to a string that contains the name of a server; for example, "QA_server".

Related Functions

TrkInitProjectListEx

TrkProjectLogout
TrkProjectLogin
TrkGetIniFile
TrkSetIniFile

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login
Tracker Administrator's Guide	Databases, supported

TrkProjectLogout

This function logs the user out of a Tracker project. Call this function after you have completed all operations associated with a project login.

C/C++

```
TRK_UINT TrkProjectLogout(
    TRK_HANDLE trkHandle) /* Input */
```

Basic

```
Declare Function TrkProjectLogout& Lib "TRKTOOLN"
    (ByVal trkHandle&)
```

Parameters

trkHandle Handle returned by **TrkHandleAlloc**.

Related Functions

TrkProjectLogin

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login

TrkQueryInitRecordList

This function executes the specified Query and returns the resulting records in the form of an in-memory record list.

The *transactionID* parameter allows you to specify a starting database event number for the records in the list. For example, if you pass the integer 36 to the *transactionID* parameter, only records that have been modified in database transactions with event numbers later than 36 will be listed.

Optionally, you can retrieve a Transaction ID for the most recent change made to the database by way of the *pNewTransactionID* parameter.

C/C++

```
TRK_UINT TrkQueryInitRecordList(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR queryName,   /* Input */
    TRK_TRANSACTION_ID transactionID,/* Input (optional) */
    TRK_TRANSACTION_ID FAR *pNewTransactionID)/* Output (optional) */
```

Basic

```
Declare Function TrkQueryInitRecordList& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal queryName$, ByVal transactionID&,
    pNewTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
queryName	Pointer to a string containing the name of the Query you want to execute.
transactionID	The starting database event number for the list.
pNewTransactionID	Pointer to the buffer that is to receive the Transaction ID for the most recent change to the database.

Related Functions

TrkInTrayInitRecordList
TrkGetNextRecord

Related Topics For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Lists and Sequential Access
Tracker User's Guide	Running and Managing Queries

TrkRecordCancelTransaction

Call this function to cancel an update or submit record transaction at any time.

C/C++

```
TRK_UINT TrkRecordCancelTransaction(
    TRK_RECORD_HANDLE trkRecordHandle)/* Input */
```

Basic

```
Declare Function TrkRecordCancelTransaction& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&)
```

Parameters

trkRecordHandle Handle returned by **TrkRecordHandleAlloc**.

Related Functions

TrkNewRecordBegin
TrkNewRecordCommit
TrkUpdateRecordBegin
TrkUpdateRecordCommit

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions

TrkRecordHandleAlloc

This function allocates a Record Handle, which is a required parameter for all Tracker functions that operate on record data. The handle you obtain remains open until you call **TrkRecordHandleFree**. If desired, you may use multiple Record Handles concurrently.

C/C++

```
TRK_UINT TrkRecordHandleAlloc(
    TRK_HANDLE trkHandle, /* Input */
    TRK_RECORD_HANDLE FAR *pTrkRecordHandle)/* Output */
```

Basic

```
Declare Function TrkRecordHandleAlloc& Lib "TRKTOOLN"
    (ByVal trkHandle&, pTrkRecordHandle&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
pTrkRecordHandle	Pointer to the variable that is to receive the Record Handle.

Related Functions

TrkRecordHandleFree

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles, Record Transactions

TrkRecordHandleFree

This function closes the specified Record Handle and releases all the memory associated with it. Call this function when you have completed all operations related to the specified Record Handle.

C/C++

```
TRK_UINT TrkRecordHandleFree(
    TRK_RECORD_HANDLE FAR *pTrkRecordHandle)/* Input/Output */
```

Basic

```
Declare Function TrkRecordHandleFree& Lib "TRKTOOLN"
    (pTrkRecordHandle&)
```

Parameters

pTrkRecordHandle Handle returned by **TrkRecordHandleAlloc**.

Related Functions

TrkRecordHandleAlloc

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Handles, Record Transactions

TrkRegisterCallback

This function registers a callback function for use in event-handling. Each callback function you register is included in a list of functions maintained by the Tracker Toolkit. To remove a function from the list of callback functions, call **TrkUnregisterCallback**.

C/C++

```
TRK_UINT TrkRegisterCallback(
    TRK_HANDLE trkHandle, /* Input */
    PTRKCALLBACK pCallbackFunction,/* Input */
    TRK_LIST_LINK_ORDER linkOrder,/* Input */
    TRK_UINT userData)    /* Input (optional) */
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
pCallbackFunction	Pointer to the callback function you want to register.
linkOrder	Possible values are `TRK_LIST_ADD_HEAD` and `TRK_LIST_ADD_TAIL`. Use `TRK_LIST_ADD_HEAD` to indicate that you want this function to be called first. Use `TRK_LIST_ADD_TAIL` to indicate that you want this function to be called last.
userData	Optional value passed to the callback function.

Callback Function Prototype

You must use the following prototype when you write callback functions to be used by the Toolkit. The value passed to each parameter depends on the event that has occurred. Events tracked by the Toolkit for callback functions are listed by callback message in the next section.

The callback functions you write should return a value of TRK_MESSAGE_HANDLED or TRK_MSG_NOT_HANDLED. If a callback function returns a value of TRK_MSG_NOT_HANDLED, the Toolkit will execute the next function in the list of registered callbacks. If the function returns a value of TRK_MSG_HANDLED, the Toolkit will call no more callback functions.

Syntax

```
TRKAPI LOADDS myCallback(
    TRK_UINT msg,         /* Input */
    TRK_UINT param1,      /* Input */
    TRK_UINT param2,      /* Input */
    TRK_CSTR strParam,    /* Input */
    TRK_UINT userData)    /* Input */
```

Parameters

msg	Message passed to the callback function. Messages are described in the following section.
param1	Long integer value passed to the callback function.
param2	Long integer value passed to the callback function.
strParam	String value passed to the callback function.
userData	Value specified when the callback function was registered using **TrkRegisterCallback**.

Callback Messages

The following callback messages and associated values are issued by the Toolkit:

TRK_MSG_API_TRACE

This message is issued whenever a Tracker Toolkit function exits. The following parameters are passed to the callback function:

msg	TRK_MSG_API_TRACE

param1 Return value of the called Tracker function.

strParam Pointer to a buffer that is to receive the name of the called Tracker Function.

Example

The following example shows the use of a callback function that logs errors occurring during Toolkit operations:

```
TRKAPI LOADDS errorHandler(
    TRK_UINT msg,
    TRK_UINT param1,
    TRK_UINT param2,
    TRK_UINT strParam,
    TRK_UINT userData)

    if (msg == TRK_MSG_API_TRACE){
        if (param1 !=TRK_SUCCESS){
        /* Log errors */
        FILE* fp = (FILE*) userData;
        fprintf(fp, "API Function '%s' returned error code %ld\n",
        strParam, param1);

    return TRK_MSG_HANDLED;

    return TRK_MSG_NOT_HANDLED;
}
.
.
.

FILE* fp = fopen("my.log", "w");
if (fp){
    rc = TrkRegisterCallback(trkHandle, &errorHandler, fp);
    if (rc != TRK_SUCCESS);
        /* Include error handling here */
        .
        .
        .
}
```

TRK_MSG_API_EXIT

This message is issued when a Tracker Handle is de-allocated using **TrkHandleFree**. The following values are passed back to the callback function:

msg TRK_MSG_API_EXIT

param1 0

TRK_MSG_ODBC_ERROR

This message is issued when the Toolkit encounters an ODBC error during operations. The following values are passed back to the callback function:

msg TRK_MSG_ODBC_ERROR

param1 Message class of the ODBC error message.

strParam ODBC error message text.

TRK_MSG_INVALID_FIELD_VALUE

This message is issued when the program tries to modify a field with an invalid value. The following values are passed back to the callback function:

msg TRK_MSG_INVALID_FIELD_VALUE

param1 The reason the field value is declared invalid. Possible values are the following:

TRK_FLDEDT_USER_NOT_IN_GROUP, which indicates that the logged in user is not assigned to the appropriate group necessary to change the field value. If this value is passed, the value of *param2* is the group ID of the user.

TRK_FLDEDT_USER_UNASSIGNED_NOT_VALID, which indicates that the logged in user is not assigned to a group.

`TRK_FLDEDT_USER_NOT_FOUND`, which indicates that the logged in user was not found in the list of users defined for the current project.

TRK_MSG_DATA_TRUNCATED

This message is issued when the Toolkit truncates string data passed to a text buffer. The following values are passed back to the callback function:

msg	`TRK_MSG_DATA_TRUNCATED`
param1	The actual length of the string.
param2	The supplied length of the string.

TRK_MSG_FORCE_LOGOUT

This message is issued when a user is forced to log out of a project by a Tracker Administrator. The following values are passed back to the callback function:

msg	`TRK_MSG_FORCE_LOGOUT`
param1	Time in minutes until logout.

TRK_MSG_IMPORT_ERROR

This message is issued when the Toolkit encounters an error during the execution of **TrkImportInit** or **TrkImportNewRecords**. The following values are passed back to the callback function:

msg	`TRK_MSG_IMPORT_ERROR`
strparam	Formatted error message.

TRK_MSG_INTRAY_PROGRESS

This message is issued while the Toolkit retrieves records from the user's In Tray. You can use this function to create a progress bar for operations conducted using **TrkInTrayInitRecordList**. The following values are passed back to the callback function:

msg	`TRK_MSG_INTRAY_PROGRESS`
param1	Total number of items to be processed.
param2	Number of items processed so far.

TRK_MSG_QUERY_PROGRESS

This message is issued while the Toolkit executes a stored Query and returns a list of records. You can use this function to create a progress bar for operations conducted using TrkQueryInitRecordList. The following values are passed back to the callback function:

msg	`TRK_MSG_QUERY_PROGRESS`
param1	Total number of items to be processed.
param2	Total number of items so far.

TRK_MSG_IMPORT_PROGRESS

This message is issued while the Toolkit imports records from a text file. You can use this function to create a progress bar for the import session. The following values are passed back to the callback function:

msg	`TRK_MSG_IMPORT_PROGRESS`
param1	Total number of items to be processed.
param2	Number of items processed so far.

TrkSetAssociationModuleName

This function sets the name of the Module you want to link to an issue with the current module association.

C/C++

```
TRK_UINT TrkSetAssociationModuleName(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_CSTR moduleName)  /* Input */
```

Basic

```
Declare Function TrkSetAssociationModuleName& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal moduleName$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
moduleName	Pointer to a string containing the Module name you want to set.

Related Functions

TrkGetAssociationModuleName
TrkGetAssociationText
TrkGetAssociationUser

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetAssociationRevisionFixed

This function sets the revision number or version label representing when the defect reported by the issue was fixed.

C/C++

```
TRK_UINT TrkSetAssociationRevisionFixed(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_CSTR revisionFixed)/* Input */
```

Basic

```
Declare Function TrkSetAssociationRevisionFixed& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal revisionFixed$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
revisionFixed	Pointer to a string containing the revision number or version label for the stage at which defect was fixed.

Related Functions

TrkGetAssociationRevisionFixed
TrkSetAssociationRevisionFound
TrkSetAssociationTimeFixed

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetAssociationRevisionFound

This function sets the revision number or version label representing when the defect reported by the issue was found.

C/C++

```
TRK_UINT TrkSetAssociationRevisionFound(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_CSTR revisionFound)/* Input */
```

Basic

```
Declare TrkSetAssociationRevisionFound& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal revisionFound$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
revisionFound	Pointer to a string containing the revision number or Version label for the stage at which the defect was found.

Related Functions

TrkGetAssociationRevisionFound
TrkSetAssociationRevisionFixed
TrkSetAssociationTimeFound

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetAssociationText

This function sets the change description of the current in-memory module association. You may call this function multiple times to set a description that is longer that the *currentBufferSize* buffer.

C/C++

```
TRK_UINT TrkSetAssociationText(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_UINT currentBufferSize,/* Input */
    TRK_CSTR description, /* Input */
    TRK_UINT dataRemaining)/* Input */
```

Basic

```
Declare Function TrkSetAssociationText& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal currentBufferSize&,
    ByVal description$, ByVal dataRemaining$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandle**.
currentBufferSize	Length of the *description* buffer.
description	Pointer to a string containing the description you want to set for the current module association.
dataRemaining	The length of data remaining to be set to the description.

Related Functions

TrkGetAssociationText
TrkGetAssociationTextLength
TrkSetAssociationUser

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetAssociationTimeFixed

This function sets the timestamp representing when the defect reported by the issue was fixed.

C/C++

```
TRK_UINT TrkSetAssociationTimeFixed(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_TIME timeFixed)   /* Input */
```

Basic

```
Declare Function TrkSetAssociationTimeFixed& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal timeFixed&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
timeFixed	Timestamp representing the time at which the reported defect was fixed.

Related Functions

TrkGetAssociationTimeFixed
TrkSetAssociationTimeFound
TrkSetAssociationRevisionFixed

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetAssociationTimeFound

This function sets the timestamp representing when the defect reported by the issue was found.

C/C++

```
TRK_UINT TrkSetAssociationTimeFound(
    TRK_ASSOC_HANDLE trkAssociationHandle,/* Input */
    TRK_TIME timeFound)   /* Input */
```

Basic

```
Declare Function TrkSetAssociationTimeFound& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal timeFound&)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
timeFound	Timestamp representing the time at which the reported defect was found.

Related Functions

TrkGetAssociationTimeFound
TrkSetAssociationTimeFixed
TrkSetAssociationRevisionFound

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetAssociationUser

This function sets the name of the user who created the current module association.

C/C++

```
TRK_UINT TrkSetAssociationUser(
    TRK_ASSOC_HANDLE trkAssociationHandle/* Input */
    TRK_CSTR userName)    /* Input */
```

Basic

```
Declare Function TrkSetAssociationUser& Lib "TRKTOOLN"
    (ByVal trkAssociationHandle&, ByVal userName$)
```

Parameters

trkAssociationHandle	Handle returned by **TrkAssociationHandleAlloc**.
userName	Pointer to a string containing the name of the user you want to set as the creator of the module association.

Related Functions

TrkSetAssociationText
TrkSetAssociationTextLength

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Chapter 9, "Integrating Tracker with Version Control"
Tracker Toolkit Reference Guide	Module Associations

TrkSetDescriptionData

This function sets the description of the current record. You may call this function multiple times to set description data that exceeds the size of the *data* buffer.

C/C++

```
TRK_UINT TrkSetDescriptionData(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_UINT currentBufferSize,/* Input */
    TRK_CSTR data,        /* Input */
    TRK_UINT dataRemaining)/* Input */
```

Basic

```
Declare Function TrkSetDescriptionData& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal currentBufferSize&, ByVal data$,
    ByVal dataRemaining&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
currentBufferSize	Length of the *data* string.
data	Pointer to a string containing the data you want to set as the description.
dataRemaining	Length of data remaining to be set in the description.

Related Functions

TrkGetDescriptionData
TrkGetDescriptionDataLength

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Descriptions

TrkSetIniFile

This function sets the current .INI file. The .INI file is used by the Tracker Toolkit to provide default login information for **TrkProjectLogin**. If no .INI file is set, the default file is used.

You may call this function before project login.

C/C++

```
TRK_UINT TrkSetIniFile(
    TRK_HANDLE trkHandle, /* Input */
    TRK_CSTR filename)    /* Input */
```

Basic

```
Declare Function TrkSetIniFile& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal filename$)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
filename	Pointer to a string containing the name and path of the .INI file you want to set for project login.

Related Functions

TrkGetIniFile
TrkProjectLogin
TrkInitProjectList

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Project Login

TrkSetNoteAuthor

This function sets the name of the author of the current in-memory note.

C/C++

```
TRK_UINT TrkSetNoteAuthor(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_STR authorName)   /* Input */
```

Basic

```
Declare Function TrkSetNoteAuthor& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal authorName$)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
authorName	Pointer to a string containing the name of the author.

Related Functions

TrkSetNoteData
TrkSetNoteTitle
TrkSetNoteCreateTime
TrkSetNoteModifyTime

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkSetNoteCreateTime

This function sets the timestamp representing the time the current note was created.

C/C++

```
TRK_UINT TrkSetNoteCreateTime(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_TIME createTime)  /* Input */
```

Basic

```
Declare Function TrkSetNoteCreateTime& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal createTime&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
createTime	Timestamp representing the time the current note was created.

Related Functions

GetNoteCreateTime
SetNoteModifyTime

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkSetNoteData

This function sets the data content of the current in-memory note.

C/C++

```
TRK_UINT TrkSetNoteData(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_UINT currentBufferSize,/* Input */
    TRK_CSTR noteData,    /* Input */
    TRK_UINT dataRemaining)/* Input */
```

Basic

```
Declare Function TrkSetNoteData& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal currentBufferSize&, ByVal noteData$,
    ByVal dataRemaining&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
currentBufferSize	Length of the *noteData* buffer.
noteData	Pointer to a string containing the data that you want to set as the content of the note.
dataRemaining	Length of remaining data to be set to the note.

Related Functions

TrkGetNoteData
TrkGetNoteDataLength
TrkSetNoteTitle
TrkSetNoteAuthor

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkSetNoteModifyTime

This function sets the timestamp for the time the current note was most recently modified.

C/C++

```
TRK_UINT TrkSetNoteModifyTime(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_TIME modifyTime)  /* Input */
```

Basic

```
Declare Function TrkSetNoteModifyTime& Lib "TRKTOOLN"
    (ByVal trkNoteHandle&, ByVal modifyTime&)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
modifyTime	Timestamp representing the time the current note was most recently modified.

Related Functions

TrkGetNoteModifyTime
TrkSetNoteCreateTime

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkSetNoteTitle

This function sets the Title of the current in-memory note.

C/C++

```
TRK_UINT TrkSetNoteTitle(
    TRK_NOTE_HANDLE trkNoteHandle,/* Input */
    TRK_CSTR noteTitle)   /* Input */
```

Basic

```
Declare Function TrkSetNoteTitle& Lib "TRKTOOLN"
     (ByVal trkNoteHandle&, ByVal noteTitle$)
```

Parameters

trkNoteHandle	Handle returned by **TrkNoteHandleAlloc**.
noteTitle	Pointer to a string containing the Title you want to set for the note.

Related Functions

TrkGetNoteTitle
TrkSetNoteAuthor

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker User's Guide	Notes
Tracker Toolkit Reference Guide	Notes

TrkSetNumericAttribute

This function stores an (ID, value) pair. Attributes with ID values of less than 1000 are used by Tracker and may be set to perform the special operations described below. You may set attributes with values of greater than 1000 to be used for any purpose.

C/C++

```
TRK_UINT TrkSetNumericAttribute(
    TRK_HANDLE trkHandle, /* Input */
    TRK_UINT attributeID, /* Input */
    TRK_UINT value)       /* Input */
```

Basic

```
Declare Function TrkSetNumericAttribute& Lib "TRKTOOLN"
    (ByVal trkHandle&, ByVal attributeID&, ByVal value&)
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
attributeID	The ID component of the pair.
value	The value component of the pair.

Special Operations

You may set the value of one of the following attribute IDs to a non-zero number in order to perform its associated operation:

TRK_CANCEL_INTRAY	Cancels record retrieval from a user's In Tray.
TRK_CANCEL_QUERY	Cancels the execution of a Query.
TRK_CANCEL_IMPORT	Cancels an import session.

Related Functions **TrkGetNumericAttribute**

TrkSetNumericFieldValue

This function sets the numeric value of the specified field in the current record.

C/C++

```
TRK_UINT TrkSetNumericFieldValue(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_UINT fieldValue)  /* Input */
```

Basic

```
Declare Function TrkSetNumericFieldValue& Lib "TRKTOOLN"
     (ByVal trkRecordHandle&, ByVal fieldName$, ByVal fieldValue&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldName	Pointer to a string that contains the name of the field whose numeric value you want to set.
fieldValue	Integer value to which you want to set the specified field.

Related Functions

TrkGetNumericFieldValue
TrkGetStringFieldValue
TrkSetStringFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions, Fields

TrkSetStringFieldValue

This function sets the string value of the specified field in the current record.

C/C++

```
TRK_UINT TrkSetStringFieldValue(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_CSTR fieldName,   /* Input */
    TRK_CSTR fieldValue)  /* Input */
```

Basic

```
Declare Function TrkSetStringFieldValue& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, ByVal fieldName$, ByVal fieldValue$)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
fieldName	Pointer to a string that contains the name of the field whose string value you want to set.
FieldValue	Pointer to a string value to which you want to set the specified field.

Related Functions

TrkGetNumericFieldValue
TrkGetStringFieldValue
TrkSetNumericFieldValue

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions, Fields

TrkUnregisterCallback

This function removes a callback function from the list of functions maintained by the Tracker Toolkit.

C/C++

```
TRK_UINT TrkRegisterCallback(
    TRK_HANDLE trkHandle, /* Input */
    PTRKCALLBACK pCallbackFunction)/* Input */
```

Parameters

trkHandle	Handle returned by **TrkHandleAlloc**.
pCallbackFunction	Pointer to the callback function you want to remove from the list.

Related Functions

TrkRegisterCallback

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Callback Functions

TrkUpdateRecordBegin

This function populates all the fields in the current record with information from the database and locks the record in the database to prevent concurrent access. Call this function to begin a record transaction that updates an existing record. When you want to update the record to the database, call **TrkUpdateRecordCommit** to end the transaction.

C/C++

```
TRK_UINT TrkUpdateRecordBegin(
    TRK_RECORD_HANDLE trkRecordHandle)/* Input */
```

Basic

```
Declare Function TrkUpdateRecordBegin& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&)
```

Parameters

trkRecordHandle Handle returned by **TrkRecordHandleAlloc**.

Related Functions

TrkUpdateRecordCommit
TrkNewRecordBegin
TrkNewRecordCommit
TrkRecordCancelTransaction

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions

TrkUpdateRecordCommit

This function updates the current record to the database and ends the record transaction. It also returns a Transaction ID for the update. To use this function, you must first call **TrkUpdateRecordBegin** initialize the record update feature.

C/C++

```
TRK_UINT TrkUpdateRecordCommit(
    TRK_RECORD_HANDLE trkRecordHandle,/* Input */
    TRK_TRANSACTION_ID FAR * pNewTransactionID)/* Output */
```

Basic

```
Declare Function TrkUpdateRecordCommit& Lib "TRKTOOLN"
    (ByVal trkRecordHandle&, pNewTransactionID&)
```

Parameters

trkRecordHandle	Handle returned by **TrkRecordHandleAlloc**.
pNewTransactionID	Pointer to the buffer that is to receive the new Transaction ID.

Related Functions

TrkUpdateRecordBegin
TrkNewRecordBegin
TrkNewRecordCommit
TrkRecordCancelTransaction

Related Topics

For more information, refer to the following:

Manual	Topic
Tracker Toolkit Reference Guide	Record Transactions

4 Return Values

In This Chapter

For information about…	See page…
Return Values	210

Return Values

The following table lists the values returned by Tracker Toolkit functions. These values are defined in TRKTOOL.H.

Return value	Description
TRK_SUCCESS	Function executed successfully
TRK_E_PARTIAL_SUCCESS	Function executed with partial success
TRK_E_VERSION_MISMATCH	Invalid Tracker Toolkit version
TRK_E_OUT_OF_MEMORY	Out of memory
TRK_E_BAD_HANDLE	Bad Toolkit handle
TRK_E_BAD_INPUT_POINTER	Bad input pointer
TRK_E_BAD_INPUT_VALUE	Bad input value
TRK_E_DATA_TRUNCATED	Data truncated
TRK_E_NO_MORE_DATA	No more data is available
TRK_E_LIST_NOT_INITIALIZED	List not initialized
TRK_E_END_OF_LIST	End of list had been reached
TRK_E_NOT_LOGGED_IN	Not logged in
TRK_E_SERVER_NOT_PREPARED	Server has not yet been prepared; use Tracker Administrator to prepare the server
TRK_E_BAD_DATABASE_VERSION	Bad database version
TRK_E_UNABLE_TO_CONNECT	Unable to connect to database server
TRK_E_UNABLE_TO_DISCONNECT	Unable to disconnect from database server
TRK_E_NO_DATA_SOURCES	No data sources are available
TRK_E_NO_PROJECTS	Database contains no valid Tracker projects
TRK_E_WRITE_FAILED	Write to database failed
TRK_E_PERMISSION_DENIED	Permission denied

Return value	Description
TRK_E_SET_FIELD_DENIED	Insufficient permission to set field
TRK_E_ITEM_NOT_FOUND	Item not found
TRK_E_CANNOT_ACCESS_DATABASE	Cannot access database
TRK_E_CANNOT_ACCESS_QUERY	Cannot access query
TRK_E_CANNOT_ACCESS_INTRAY	Cannot access In Tray
TRK_E_CANNOT_OPEN_FILE	Unable to open file
TRK_E_INVALID_RECORD_TYPE	Invalid record type
TRK_E_INVALID_FIELD	Invalid field
TRK_E_INVALID_CHOICE	Invalid choice
TRK_E_INVALID_USER	Invalid user
TRK_E_INVALID_SUBMITTER	Invalid submitter
TRK_E_INVALID_OWNER	Invalid owner
TRK_E_INVALID_DATE	Invalid date
TRK_E_INVALID_STORED_QUERY	Invalid stored query name
TRK_E_INVALID_MODE	Invalid mode
TRK_E_NO_CURRENT_ATTACHED_FILE	No current attached file
TRK_E_NO_CURRENT_ASSOCIATION	No current association
TRK_E_NO_RECORD_BEGIN	No current record
TRK_E_NO_MODULE	No current module association
TRKEXP_E_EXPORT_WRONG_VERSION	Import file generated with a different version of Tracker and is not compatible
TRKEXP_E_EXPORTSET_NOT_INIT	Export set not initialized
TRKEXP_E_NO_EXPSET_NAME	No export set name
TRKEXP_E_BAD_EXPSET_NAME	Bad export set name
TRKEXP_E_EXPSET_FAIL_CREATE	Failed to create export set
TRKEXP_E_IMPORTMAP_NOT_INIT	Import map not initialized

Return value	Description
TRKEXP_E_NO_IMPMAP_NAME	No import map name
TRKEXP_E_BAD_IMPMAP_NAME	Bad import map name
TRKEXP_E_IMPMAP_FAIL_CREATE	Failed to create import map
TRKEXP_E_IMP_VALIDATE_FAIL	Invalid import file; try selecting different format
TRKEXP_E_USER_NOEXIST	User does not exist; set the option to add new users
TRKEXP_E_USER_ADD	Failed to add new user
TRKEXP_E_IMPORT_NOT_INIT	Import not initialized
TRKEXP_E_BAD_EMBEDDED_QUOTE_ARG	Invalid embedded quote type
TRKEXP_E_BAD_DATEFORMAT_ARG	Invalid date format
TRKEXP_E_BAD_TIMEFORMAT_ARG	Invalid time format
TRKEXP_E_BAD_CHOICE_OPTION_ARG	Invalid choice option
TRKEXP_E_BAD_USER_OPTION_ARG	Invalid user option
TRKEXP_E_BAD_NUMBER_OPTION_ARG	Invalid number option
TRKEXP_E_BAD_DATE_OPTION_ARG	Invalid date option
TRKEXP_E_ALL_NOTES_SELECTED	Cannot add a note to the export set when *all notes* is selected
TRKEXP_E_READ_EXPORTHDR	Failed to read import file header
TRKEXP_E_WRITE_EXPORTHDR	Failed to write export header
TRKEXP_E_READ_RECORDHDR	Failed to read record header
TRKEXP_E_WRITE_RECORDHDR	Failed to write record header
TRKEXP_E_WRITE_FIELD	Failed to write field to the export file
TRKEXP_E_OPEN_FILE	Unable to open import file
TRKEXP_E_READ_FIELD	Unable to read field from the import file
TRKEXP_E_READ_FIELD_WRONG_TYPE	Field type mismatch in the import file
TRKEXP_E_BAD_ITEM_TYPE	Bad export set item type

Return value	Description
TRKEXP_E_WRITE_TO_DB	Failed to write field to the database
TRKEXP_E_READ_FROM_DB	Failed to read field from the database
TRKEXP_E_BAD_DATE	Invalid date
TRKEXP_E_BAD_CHOICE	Invalid choice
TRKEXP_E_BAD_NUMBER	Invalid number
TRKEXP_E_OPEN_ERRORLOG	Unable to open import error log file
TRKEXP_E_BAD_ERRORLOG_PATH	Invalid import error log path
TRKEXP_E_LOGGING_ERROR	Failed to write to import error log file
TRKEXP_E_CLOSE_ERRORLOG	Unable to close the import error log file
TRKEXP_E_IMPORT_PERMISSION	Permission denied for import
TRKEXP_E_EXPORT_PERMISSION	Permission denied for export
TRKEXP_E_NEW_USER_PERMISSION	Permission denied for importing new users
TRKEXP_E_NEWCHOICE_SYSFIELD	Cannot add new choice to system field
TRKEXP_E_EXTRA_FIELDS	Extra fields encountered that are not specified by the field names on the first line

Appendix A: Glossary of Tracker Toolkit Terms

The following glossary provides descriptions of some of the terms used in this manual. The *Tracker User's Guide* provides other descriptions of terms that may be helpful in using the Toolkit.

access rights Permission information on a field or a description of a record. Access rights depend on the identity of the logged in user, and whether the record is to be submitted or updated to the database. Possible access rights are Read Only and Read and Write, represented in the Toolkit by the constants `TRK_READ_ONLY` and `TRK_READ_WRITE`.

attached file A binary or ASCII file attached to an issue.

change description Text in a module association describing the changes made to a workfile.

description Text describing an issue; there is only one description associated with each record.

DBMS Name The name and path of the DBMS (database management system) server used by Tracker. As a parameter in Tracker functions, this is a string; for example, `m:\merant\tracker\db`.

DBMS Type An ODBC database management system used by a Tracker database. Tracker currently supports Microsoft SQL Server, MSDE, Sybase, and Oracle. As a parameter in Tracker functions, this is a string; for example, `Oracle`.

data storage mode The format of an attached file. The possible data storage modes are *binary* and *ASCII*, represented in the Toolkit by the constants `TRK_FILE_BINARY` and `TRK_FILE_ASCII`.

field type A value used by the Toolkit to represent the type of a field.

handle A value that refers to state information managed internally by the Toolkit.

module association A set of data that describes a link between an issue and a module in a Version Manager archive.

note Text attached to a record that provides additional information about an issue.

record type A value used by the Toolkit to represent the type of a record.

submit When you *submit* a record, you are posting an entirely new record to a Tracker database.

Timestamp A value representing the number of elapsed seconds since midnight, Jan. 1, 1970.

Transaction ID A value representing an event in a Tracker database. Each change made to a record in the database receives a Transaction ID. Transaction IDs are allocated sequentially by record type, so each such value represents the *nth* event to occur with a record of a specific type.

update When you *update* a record, you are modifying a pre-existing record in a Tracker database.

Version ID The version number of the Tracker Toolkit, represented by the constant `TRK_VERSION_ID`. The Version ID of the Toolkit must match the version number of the Tracker application in order for any Tracker functions to be used.

Index

A

access rights 215
attached files 41, 215
    adding 42
    deleting 42
    extracting 42

C

callback functions 36
change descriptions 215
contacting technical support 13
conventions, typographical 12
current items 28

D

data storage mode 216
DBMS Name 215
DBMS Type 215
descriptions 40, 215
documentation, ordering hard-copy manuals 13

E

error codes 210
exporting records 32

F

field types 39, 216
fields 38
    modifying values 40
    retrieving values 40

H

handles 23, 216
hard-copy manuals, ordering 13

I

importing records 34
issue types 40
    invalid fields value 41
    retrieving 40
    setting 40

L

lists, in-memory 28

M

Merant, contacting 13
module associations 43, 216
    adding 43
    deleting 43
    modifying 43
    retrieving 43

N

notes 41, 216
    adding 41
    deleting 41
    modifying 41
    retrieving 41

O

online help
    accessing 12
    for the command-line interface 12
    for the GUI 12
ordering hard-copy manuals 13

P

printed manuals, ordering 13
project login and logout 25

Q

Queries, executing 30

R

record transactions 30
record types 38, 216
records 38
    exporting 32
    importing 34
    retrieving 29
    submitting 30
    updating 30
return values 210

S

sample programs 44
sequential access 28

T

Timestamps 216
Transaction IDs 216
TrkAddNewAssociation 58
TrkAddNewAttachedFile 59
TrkAddNewNote 60
TrkAssociationHandleAlloc 61
TrkAssociationHandleFree 62
TrkAttachedFileHandleAlloc 63
TrkAttachedFileHandleFree 64
TrkCloneRecordBegin 65
TrkCloneRecordCommit 66
TrkDeleteAssociation 67
TrkDeleteAttachedFile 68
TrkDeleteNote 69
TrkDeleteRecord 70
TrkExportHandleAlloc 71
TrkExportHandleFree 72
TrkExportOneRecord 73
TrkExportRecordsBegin 74
TrkExportRecordsClose 77

TrkExtractAttachedFile 78
TrkGetAssociationModuleName 79
TrkGetAssociationRevisionFixed 80
TrkGetAssociationRevisionFound 81
TrkGetAssociationText 82
TrkGetAssociationTextLength 83
TrkGetAssociationTimeFixed 84
TrkGetAssociationTimeFound 85
TrkGetAssociationTransactionID 86
TrkGetAssociationUser 87
TrkGetAttachedFileName 88
TrkGetAttachedFileStorageMode 89
TrkGetAttachedFileTime 90
TrkGetAttachedFileTransactionID 91
TrkGetDescriptionAccessRights 92
TrkGetDescriptionData 93
TrkGetDescriptionDataLength 94
TrkGetDescriptionTransactionID 95
TrkGetFieldAccessRights 96
TrkGetFieldDefaultNumericValue 97
TrkGetFieldDefaultStringValue 98
TrkGetFieldMaxDataLength 99
TrkGetFieldRange 101
TrkGetFieldTransactionID 103
TrkGetFieldType 104
TrkGetIniFile 106
TrkGetLoginDataSource 107
TrkGetLoginDBMSName 107
TrkGetLoginDBMSType 108
TrkGetLoginProjectName 109
TrkGetLoginUserName 110
TrkGetNextAssociation 111
TrkGetNextAttachedFile 112
TrkGetNextChoice 113
TrkGetNextDBMSType 114
TrkGetNextField 115
TrkGetNextIssueType 116
TrkGetNextNote 117
TrkGetNextProject 118
TrkGetNextQueryName 119
TrkGetNextRecord 120
TrkGetNextRecordType 121
TrkGetNextUser 122
TrkGetNoteAuthor 123
TrkGetNoteCreateTime 124
TrkGetNoteData 125
TrkGetNoteDataLength 126
TrkGetNoteModifyTime 127
TrkGetNoteTitle 128
TrkGetNoteTransactionID 129
TrkGetNumericAttribute 130
TrkGetNumericFieldValue 131
TrkGetProjectTransactionID 132
TrkGetQueryRecordType 133
TrkGetRecordRecordType 134
TrkGetRecordTransactionID 135
TrkGetRecordTypeName 136
TrkGetSingleRecord 137
TrkGetStringFieldValue 138
TrkGetUserEmail 139
TrkGetUserFullName 140
TrkHandleAlloc 141
TrkHandleFree 142
TrkImportHandleAlloc 143
TrkImportHandleFree 144
TrkImportInit 145
TrkImportNewRecords 148
TrkImportSetOptions 149
TrkInitAssociationList 152
TrkInitAttachedFileList 153
TrkInitChoiceList 154
TrkInitDBMSTypeList 155
TrkInitFieldList 156
TrkInitIssueTypeList 157
TrkInitNoteList 158
TrkInitProjectList 159
TrkInitProjectListEx 161
TrkInitQueryNameList 162
TrkInitRecordTypeList 163
TrkInitUserList 164
TrkInTrayInitRecordList 165
TrkModifyIssueType 167
TrkNewRecordBegin 168
TrkNewRecordBeginEx 169
TrkNewRecordCommit 170
TrkNoteHandleAlloc 171
TrkNoteHandleFree 172
TrkProjectLogin 173

TrkProjectLoginEx 175
TrkProjectLogout 177
TrkQueryInitRecordList 178
TrkRecordCancelTransaction 180
TrkRecordHandleAlloc 181
TrkRecordHandleFree 182
TrkRegisterCallback 183
TrkSetAssociationModuleName 189
TrkSetAssociationRevisionFixed 190
TrkSetAssociationRevisionFound 191
TrkSetAssociationText 192
TrkSetAssociationTimeFixed 193
TrkSetAssociationTimeFound 194
TrkSetAssociationUser 195
TrkSetDescriptionData 196
TrkSetIniFile 197
TrkSetNoteAuthor 198
TrkSetNoteCreateTime 199
TrkSetNoteData 200
TrkSetNoteModifyTime 201
TrkSetNoteTitle 202
TrkSetNumericAttribute 203
TrkSetNumericFieldValue 204
TrkSetStringFieldValue 205
TrkUnregisterCallback 206
TrkUpdateRecordBegin 207
TrkUpdateRecordCommit 208
typographical conventions 12

V

Version ID 216