ORACLE
INSURANCE

Oracle Insurance Compliance Tracker

Technical Guide

version 6.6

Part number: E15016-01

November 2009

Copyright

Primary Authors: Andrew Brooke and Ken Weinberg

CONTENTS

Chapter 1

Introduction

Welcome to the Tracker 6.6 Technical Guide.

This guide is for advanced users such as system and database administrators who need to set up and maintain Tracker.

Note: For installation procedures, and hardware and software requirements, please see the *Tracker 6.6 Installation Guide*.

This chapter describes:

Tracker

Tracker is designed to integrate with IStream Document Manager and Launcher to offer you collaborative capabilities across your organization. The integration of these other products with Tracker will allow you to seamlessly manage projects within your organization from the time of initial document creation through the filing process to actual implementation. It helps you create your filings from a centralized point. All content of the filings can be created, maintained and updated within Tracker, letting you focus on generating content. By using Launcher workflows to manage the filing process, you can develop filings knowing that you are not missing any steps required by your organization.

Furthermore, users who are at the appropriate security level can follow the stages of the filing process and know who has been assigned the tasks involved in taking the filing package to market.

If you are not integrating with IStream Document Manager for some or all of your filings, then Tracker can be used to create and manage filings on its own. This gives you a flexible solution, based upon your business needs.

Document Conventions

Tips, Notes, Important Notes and Warnings

Tip: A **Tip** provides a better way to use the software.

Note: A **Note** contains special information and reminders.

Important: An **Important** note contains significant information about the use and understanding of the software.

Warning: A **Warning** contains critical information that if ignored, may cause errors or result in the loss of information.

Other Document Conventions

- Microsoft Window names, buttons, tabs and other screen elements are in bold, for example: Click **Next**.
- paths, URLs and code samples are in the Courier font, for example: `C:\Windows`
- values that you need to enter or specify are indicated in the italicized Courier font, for example, *`server_name`*
- values that are optional are indicated with square brackets, for example [`reserved`]

Tracker Documentation

Tracker includes the following documents and online help files. If you need a copy of any of these documents, please contact your system or product administrator.

- The *Tracker User Guide* contains overviews, step-by-step procedures and descriptions of the screens and fields.
- The *Tracker Online Help* contains the same information as the User's Guide, but in an online help format with a search tool, an index and a table of contents.
- The *Tracker Release Notes* include general product information, product enhancements and new features, supported platforms and third-party software, assorted considerations, and known issues and limitations.
- The *Tracker Installation Guides* contain system requirements and detailed installation and configuration information. Guides are supplied for new installations and upgrades, and for both Oracle and SQL environments.
- The *Tracker Technical Guide* is for system administrators and includes information about the optional DMS, maintaining DMS components, log files, error levels and Tracker Monitor, technical information about the Regulatory Specialist files and validation process, and troubleshooting information.

Chapter 2

Tracker Log and Launcher Temp Files

Tracker is configured after installation to send log information to a set of log files. You can increase the error messages recorded in these log files to help troubleshoot, or if you are requested to do so by Global Customer Support. The level of detail recorded in the log files is controlled by a key in the Windows registry.

If you are using Tracker with Launcher, you can configure Tracker to keep or delete the temporary file created by Launcher.

This chapter describes:

Configuring Error Message Levels

The following levels of error messages exist in Tracker:

- ERROR
- DEBUG

The default is that Tracker will log errors at the **ERROR** level. Each type of log file will have different information recorded to it, depending on what the overall error message level is in Tracker.

Changing the Error Message Level

You can change the level of error message which is logged by editing a registry key.

Note: Back up your registry before working with it.

Method: Change the error message level

---

1. Run `regedit`.
2. Navigate to the following key:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\Tracker\6.6\
   General\LogVerboseLevel
   ```

3. If the key is not found, create it.
4. Set the value of the key to one of the following values:
   - `ERROR`
   - `DEBUG`

Note: These values are not case-sensitive, so, for example, `Error` or `Debug` would work.

5. Close `regedit`.
6. If you are changing the level on the Tracker Monitor machine, stop and restart Tracker Monitor.

   If you are changing the level on the client workstation, exit and restart Tracker.

   The new logging level is now used with Tracker.

Error Levels and Log File Contents

There are log files used with various components in the Tracker system. For each log file, a different amount of information will be recorded based on the global error logging level you have set. The default logging level, ERROR, can be changed to a more verbose logging level, DEBUG, by changing a registry setting. See *Changing the Error Message Level* on page 10.

To configure the location where the log files are saved, enter a valid folder name in this registry entry:

`HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\Tracker\6.6\General\LogDirectory`

Note: Almost all log file names end in a date to indicate when each file was created. Dates use the `yyyymmdd` format. For example, a log file called `PA_20060615.log` was created on June 15, 2006. This information helps you to decide which log files to keep and to delete.

Tracker Client

The following log files are associated with the Tracker client.

Log File Name	Area Logged	Logging level = ERROR (default level)	Logging level = DEBUG
`PA_`*`yyyymmdd`*`.log`	Document assembly	Critical errors only	Critical errors and debug messages
`PM_`*`yyyymmdd`*`.log`	Database access	Critical errors only	Critical errors and debug messages
`RSGUI_`*`yyyymmdd`*`.log`	Regulatory Specialist	Critical errors only	Critical errors and debug messages
`TkCalligo.log`	Calligo/ IStream Document Manager	Critical errors only	Critical errors and debug messages
`TRKDBUG_yyyymmdd.log`	all (if Tracker cannot determine which module created the problem)	Critical errors only	Critical errors and debug messages

Tracker Monitor

The following log files are associated with Tracker Monitor.

Log File Name	Area Logged	Logging level = ERROR (default level)	Logging level = DEBUG
`TkrSerffFiler_`*`yyyymmdd`*`.log`	Contains SERFF module log messages.	Critical errors only	Critical errors, SERFF filing submission and response messages, and debug messages.
`TkrWF_`*`yyyymmdd`*`.log`	Other Tracker Monitor messages.	Critical errors only	Critical errors and information on start and shutdown of Tracker Monitor threads, and debug messages
`PM_`*`yyyymmdd`*`.log`	Database access	Critical errors only	Critical errors and debug messages
`TRKDBUG_yyyymmdd.log`	All (used if Tracker cannot determine which module created the problem)	Critical errors only	Critical errors and debug messages
`RSCabloadApp_yyyymmdd.log`	Cab loading	Critical errors only	Critical errors and debug messages

Tracker-SERFF Web Service

The following log files are associated with the Tracker-SERFF Web service (running as a SOAP service). They are created on both the server and client systems when the logging debug level is set to DEBUG.

Log File Name	Area Logged	Logging level = ERROR	Logging level = DEBUG
`SENT_`*`yyyymmdd`*`.log`	Raw SOAP messages sent to SERFF	No information is logged	Raw SOAP messages sent to SERFF
`RECV_`*`yyyymmdd`*`.log`	Raw SOAP messages received from SERFF	No information is logged	Raw SOAP messages received from SERFF
`SOAP_`*`yyyymmdd`*`.log`	gSOAP diagnosis log	No information is logged	gSOAP diagnosis log

Configuring the Tracker Temporary File

Tracker creates a temporary file that records information about the communication between Tracker and Launcher. (Launcher is an optional, separate application that integrates with Tracker.)

You can configure Tracker to keep or delete this temporary file.

Method: Configure the Tracker temporary file

1. Run `regedit`.
2. Navigate to the following key:

   `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\Tracker\6.6\General\KeepTemporaryFile`
3. Set the value of this key as follows:
   - to *keep* the temporary file, set the key value to `1`
   - to *delete* the temporary file after it is processed, set the key value to `0`

Chapter 3

Tracker Monitor

Tracker Monitor is a separate application from the Tracker client and IStream Document Manager. It is used to:

- monitor and facilitate communications between Tracker and SERFF when users are entering and updating SERFF filings
- manage all regulatory specialist update and communication between Tracker and the regulatory specialist update server

Because of these functions, you will need to properly install and configure Tracker Monitor and ensure that the service is running.

This chapter describes:

- *Configuring Tracker Monitor* on page 16
- *Tracker Monitor Executable Files* on page 17

For information about how to install and configure Tracker Monitor, see the *Tracker Installation Guide*.

Configuring Tracker Monitor

Tracker Monitor runs as a Windows service on the Tracker file server or a dedicated server. You cannot run Tracker Monitor on a Tracker workstation.

During installation, you need to enter a user name and password that will be used for Tracker Monitor. This user can be a domain or local user, however it must belong to the local administrators group.

If the user name and password entered during installation are not valid, then the Tracker Monitor services will not be registered. You will therefore need to manually register the monitor services.

Method: Manually register the monitor services

1. In DOS, change the active directory to the one where the Tracker Monitor files are installed.
2. Run the following commands to register the Tracker Monitor services:

   ```
   C:\[.NET Framework]\installutil /username=
   [machine name\user name] /password=[password]
   /eventlogname=TrackerMonitor /eventlogsource=TkrMonitor
   tkrmonitor.exe
   ```

   ```
   C:\[.NET Framework]\installutil /username=
   [machine name\user name] /password=[password]
   /eventlogname=TrackerMonitor /eventlogsource=TkrCabLoad
   tkrcabload.exe
   ```

3. If you are using Launcher, register the `IPDLinkService` also:

   ```
   C:\[.NET Framework]\installutil /username=[machine
   name\user name] /password=[password]/eventlogname=
   IPDLinkService/eventlogsource=IPDLinkService
   IPDLinkService.exe
   ```

   For all these services:

   - `[Windows NET]` is the path in which the Microsoft Windows.NET framework is installed: this is typically: `C:\WINDOWS\Microsoft.NET\Framework\v...\`
   - *`[machine name\user name]`* and *`[password]`* are the machine name, user name and password for Tracker Monitor

Tracker Monitor Executable Files

Tracker Monitor consists of the following executable files:

- *TkrMonitor.exe* on page 17
- *TkrCabLoad.exe* on page 18
- *IPDLinkService.exe* on page 19
- *TkrWF.exe* on page 20

TkrMonitor.exe

`TkrMonitor.exe` is a .NET Windows services executable file that monitors the activities of the other two files: `TkrCabLoad.exe` and `TkrWF.exe`. At a specified time interval, it launches then checks if the other services are properly running. If not, `TkrMonitor.exe` stops and restarts them individually.

The progress of `TkrMonitor.exe` and any errors are logged to the windows event log and the Tracker database. It uses the same logging level as in other parts of Tracker: see *Tracker Log and Launcher Temp Files* on page 9 for more information about logging.

Configuring TkrMonitor.exe

To configure `TkrMonitor.exe`, you edit `TkrMonitor.exe.config`, a .NET configuration file. The `<appSettings>` section of this file contains settings of various keys, described in the following sections.

`Important:` Contact Global Customer Support. before making any changes to this file.

Key: EventLogSource

Default value: `TkrMonitor`

The name of the source that `TkrMonitor.exe` uses to log messages to the Windows event log. You may need to reboot the server to allow any changes to this key to become active.

Key: EventLogName

Default value: `TrackerMonitor`

The name of the event log to which `TkrMonitor.exe` logs messages. You may need to reboot the server to allow any changes to this key to become active.

Key: WakeupTimerIntervalInSeconds

Default value: `480`

The frequency in seconds that Tracker Monitor checks that the other executable files are still running properly. You need to restart the `TkrMonitor.exe` service for any changes to this key to become active.

Key: FirstWakeupDelayInSeconds

Default value: 120

After starting `TkrMonitor.exe` for the first time, the system will wait the number of seconds specified in `FirstWakeupDelayInSeconds` before actually launching `TkrCabLoad.exe` and `TkrWF.exe`. You need to reboot the server for any changes made to this key to become active.

TkrCabLoad.exe

`TkrCabLoad.exe` is a .NET Windows services executable file that performs all the updating processes. It is a .NET Remoting server that monitors a TCP port in the Tracker Monitor server.Tracker clients communicate with this service through this TCP port in order to perform any manual regulatory specialist updates or to query for updates. This service also starts the pre-scheduled automatic .cab file download.

Important: The firewall on the Tracker Monitor server must allow incoming network traffic at the TCP port specified in this service. For instructions about setting the port number, see *Configuring Tracker to Access the Regulatory Specialist Update Server* of the *Tracker User Guide* or *Online Help.*

The progress of `TkrCabLoad.exe` and any errors are logged to the windows event log and the Tracker database. It uses the same logging level as in other parts of Tracker: see *Tracker Log and Launcher Temp Files* on page 9 for more information.

Configuring TkrCabLoad.exe

Important: Contact Global Customer Support. before making any changes to this file. Note that you may need to reboot the server for any changes to this file to become active.

To configure `TkrCabLoad.exe`, edit `TkrCabLoad.exe.config`, a .NET configuration file.

The following section of this file controls the maximum size of the .cab file (in KB) that can be downloaded from Oracle.

```
<microsoft.web.services2>
  <messaging>
    <maxRequestLength>20480</maxRequestLength>
  </messaging>
  <diagnostics />
</microsoft.web.services2>
```

The default size (indicated above in bold) is 20480 KB.

All other settings are stored in the `<appSettings>` section of `TkrCabLoad.exe.config`, as described in the following sections.

Key: EventLogSource

Default value: `TkrCabLoad`

The name of the source that `TkrCabLoad.exe` uses to log messages to the Windows event log.

Key: EventLogName

Default value: `TrackerMonitor`

The name of the event log to which `TkrCabLoad.exe` logs messages.

Key: WakeupTimerIntervalInSeconds

Default value: `300`

The frequency in seconds that `TkrCabLoad.exe` checks for an update.

Key: FirstWakeupDelayInSeconds

Default value: `120`

After starting `TkrCabLoad.exe`, the system waits the number of seconds specified in `FirstWakeupDelayInSeconds` before actually checking for an update.

Key: LockDBWakeupTimerInMSec

Default value: `30000`

An internal system parameter. **Do not modify.**

Key: ObjectLeaseTimeInMinutes

Default value: `120`

An internal system parameter. **Do not modify.**

IPDLinkService.exe

`IPDLinkService.exe` is a .NET Windows services executable file that moves filing profiles between Tracker and Launcher.

The progress of `IPDLinkService.exe` and any errors are logged to the windows event log. It uses the same logging level as in other parts of Tracker: see *Tracker Log and Launcher Temp Files* on page 9 for more information about logging.

Configuring IPDLinkService.exe

To configure `IPDLinkService.exe`, you edit `IPDLinkService.exe.config`, a .NET configuration file. The `<appSettings>` section of this file contains settings of various keys, described in the following sections.

Important: Contact Global Customer Support. before making any changes to this file.

Key: EventLogSource

Default value: `IPDLinkService`

The name of the source that `IPDLinkService.exe` uses to log messages to the Windows event log.

Important: This key is for internal use, and should not be changed.

Key: EventLogName

Default value: `IPDLinkService`

The name of the event log to which `IPDLinkService.exe` logs messages.

Important: This key is for internal use, and should not be changed.

Key: LogToDb

Default value: `false`

This is an internal parameter to `IPDLinkService.exe` and it has to be set to `false`.

Key: WakeupTimerIntervalInSeconds

Default value: 180

The frequency in seconds that `IPDLinkService.exe` wakes up to process profile requests sent to Tracker or responses received from Tracker. You need to restart the `IPDLinkService.exe` service for any changes to this key to become active.

Key: FirstWakeupDelayInSeconds

Default value: 120

After starting `IPDLinkService.exe` for the first time, the system will wait the number of seconds specified in `FirstWakeupDelayInSeconds` before actually launching processing profile requests sent to Tracker or responses received from Tracker. You need to reboot the server for any changes made to this key to become active.

TkrWF.exe

The executable file for the previous version of Tracker Monitor. It manages the communication between the workflow and Tracker and between SERFF and Tracker. It is still a COM server, and is launched by a Windows service: `TkrMonitor.exe`.

Chapter 4

The Regulatory Specialist

This chapter describes:

- *The Regulatory Specialist Files* on page 22
- *Validating the Update* on page 24
- *The Update Progress Log* on page 27

Note: For more information about the regulatory specialist, see the *Working with the Regulatory Specialist* section in the *User Guide* or *Online Help*.

The Regulatory Specialist Files

The regulatory specialist update that is downloaded from the Tracker server consists of the following files:

Cab_x.cab

This file uses the current file structure. The file extraction utility produces the same .cab file as in previous releases. All updated model documents and the .cab configuration file (`TkrLoad.ini`) are compressed into a .cab file. However, the update period within this .INI file is now ignored in the new .cab file.

The naming syntax for the .cab file is: `Cab_`*n*`.cab`, for example, `Cab_1.cab`, `Cab_2.cab`, `Cab_3.cab`, and so on.

Readme.doc

The ReadMe Word file that describes the update.

CabInfo.xml

A new XML file containing additional information about the .cab file and the hash information.

`CabInfo.xml` has the following format:

```
<?xml version="1.0" standalone="yes"?>
<!--This file contains information about the .cab update-->
<cabinfo>
  <cabid>15</cabid>
    <nextcabid>16</nextcabid>
      <description>A short description of the .cab file</description>

      <hashtable>
        <tableitem key="RSFilingForms" count="1674" hash="425964352"/>
        <tableitem key="RSDocumentModel" count="1674" hash="124871543"/>
        <tableitem key="RSFilingFormReq-AK-BM" count="612"
hash="524357165"/>
       ...
       </hashtable>

  <hashfile>
    <fileitem key="INS00001.CMS" count="302592" hash="302592"/>
    <fileitem key="INS00042.CMS" count="54364" hash="54364"/>
       ...

  </hashfile>

</cabinfo>
```

This XML file has the following elements:

- `<cabid>` – the id of the .cab file
- `<nextcabid>` – the ID of the .cab file that will be loaded after the current cab file
- `<description>` – a description of the .cab file: this appears when users query the available .cab files.
- `<hashtable>` – the hash values of various tables after the update is applied
- `<hashfile>` – the hash values of all .CMS files after the update is applied

Validating the Update

To ensure that the Tracker database is properly updated, the systems performs a hash value check before and after the update. Only the regulatory specialist information in the Tracker database and the filing form model documents are checked.

The hash calculation has two parts:

- the database hashing: include hashing the regulatory specialist tables and the records updated by Oracle
- hashing of the filing form model documents

All hash information is stored in an XML file and downloaded from the Tracker regulatory specialist update server.

Note: Only Global Customer Support. can enable or disable the hash-checking process. Please contact them for assistance.

Table Hashing

Information about table hashing is stored in the `<hashtable>` section of the `CabInfo.XML` file. When calculating the hash, the system must exclude any information that is associated with NAIC Taylor state (code `ZZ`), because this state is not included during a regular update.

The attribute key refers to the table for which the record count and hash value applies (except for the `RSFilingFormReq` table). The attribute hash contains the hash value of the corresponding tables or sets of records. The attribute count is the total number of relevant rows in the records used in the hash calculation.

The method to determine which rows are included in the hash calculation varies with each table. The following sections list the tables being hashed and the criteria by which the rows are selected for the hashing calculation.

RSDepartmentAddresses

All rows are included in the hash calculation.

RSGeneral

All rows are included in the hash calculation.

RSFilingForms

All rows are included in the hash calculation.

RSDocumentModel

All rows are included in the hash calculation.

RSLOB

All rows are included in the hash calculation.

RSFilingFormReq

All rows are included in the hash calculation.

Because this is a large table, it is divided up by **StateCode** and **LOBCode** for the calculations. The hash and row count is calculated for each **StateCode** and **LOBCode** combination.

The attribute key is in the following format:

`RSFilingFormReq-`*`2 digit state code-3 digit LOB code`*

The following table is an example of table hash data in `CabInfo.xml`:

Key	Count	Hash
RSDepartmentAddresses	432	1978396164
RSGeneral	3375	1376734834
RSFilingForms	1674	425964352
RSDocumentModel	1674	1248791543
RSLOB	51	1519735124
RSFilingFormReq-AK-AE	241	684521793
RSFilingFormReq-AK-BM	612	524357165
RSFilingFormReq-AK-BO	214	1975354820
RSFilingFormReq-AZ-AE	352	863571025
…	…	…

RSCode

Only rows with **CategoryID** field < 5000 are hashed.

RSStateCode

Only rows with ***both*** the following conditions are hashed:

- **CategoryID** field < 5000

  - and -

- **StateCode** not equal to ‘ZZ’

File Hashing

File hashing is the hashing process for filing form model documents. Only filing forms that do not have an **obsolete date** setting and that have a **FilingFormID** less than 1,000,000,000 are included in the hash. Filing letters are not hashed.

Actual hash values for the model documents are not calculated. Instead, only the file size of all the model documents is checked. The information is stored in `CabInfo.xml`. The **Key** attribute stores the file name of the model document. The **Count** and **Hash** attributes contain the size of the file.

The following table is an example of information stored in `CabInfo.xml`:

Key	Count	Hash
INS00001.CMS	38563	38563
INS00002.CMS	302592	302592
INS00006.CMS	272896	272896
…	…	…

The Update Progress Log

The Tracker monitor logs the progress of the regulatory specialist update to a `RSLogMessages` table in the Tracker database. To troubleshoot the update, you will need access to this log. During a manual regulatory specialist update, the same progress messages that are stored in this log are displayed at the Tracker workstations.

By default, the contents of the log are never cleared. However, you should configure Tracker to clear the log after a certain number of days. For instructions, see *Configuring Ad Hoc Regulatory Specialist Updates* in the *Tracker User Guide* or *Online Help*.

Chapter 5

Troubleshooting

The following chapter deals with troubleshooting any problems encountered in your Tracker installation. The main areas relate to working with the DMS components installed with Tracker.

Tasks included in this chapter are:

Setting Log Levels

Several system logs are either generated automatically as the system runs, or can be enabled to help Global Customer Support. track and resolve any issues you might have.

Log Levels

The following log levels exist in Livelink. If a file is listed as being in both levels, then you can obtain additional log information by configuring your system appropriately.

LEVEL 1	**Initial contact**: Easily generated or already generated logs.
LEVEL 2	**Follow-up**: Some logs must be enabled by way of user intervention. Others are enabled be default however, they will not contain usable information until an error occurs.

Server Log Levels

Level 1	Level 2
• exception log • NT event viewer (application, security, system) • sysreport.txt • calver.txt (verlistm.exe)	• exception.log (proxy) • component logs • SQL connection logs • thread logs • tomcat.log • except log • NT event viewer (application, security, system) • SQL.log (ODBC client trace) • sysreport.txt • msinfo32.exe (Win98) • winmsd (Win2000, WinNT) • calver.txt (verlistm.exe)

Enabling Diagnostics on the Livelink Server

Note: Tracker works with both Calligo 5.4 and its later version, IStream Document Manager 6.3. Both Calligo and IStream Document Manager are built on Livelink. Therefore, all references to Livelink refer indirectly to Calligo and IStream Document Manager.

The following diagnostics can be enabled on the Livelink server:

- Generating a Livelink System Report
- Generating a Diagnostics Report (Level 1-5)
- Changing the Logging Level

Generating a Livelink System Report

From your Internet Browser, enter the following URL:

```
http://ServerName/ServiceName/livelink.exe?func=
admin.sysreport
```

Generating a Diagnostics Report (Level 1-5)

This report compares the contents of your External Document Storage with your internal IStream database.

1. On the **Livelink Administration** page, under the **Database Administration Section**, click the **Maintain Current Database** link.
2. Click **Verify This Database**.

Changing the Logging Level

You can change the logging level to control which level(s) of messages are displayed and logged.

Warning: This procedure involves editing the Windows registry and is therefore for system administrators only. We recommend that you back up your registry before proceeding. Also, shut down the server before making changes and restart after.

Note: The default setting is 3.

Method: To change the logging level

---

1. In the Windows registry, locate the following key:

   ```
   HKEY_LOCAL_MACHINE\SOFTWARE\Oracle\IStream_Components\
   Shared\Log\2.0
   ```

2. Set the **Logging Level** value. The following table indicates which level(s) of messages will be logged for each **Logging Level** value:

Logging Level	Info	Warning	Error	Critical
2	●	●	●	●
3		●	●	●
4			●	●
8				●
9	No messages logged.			

For example, a logging level of 4 will record **Error** and **Critical** messages only.

Livelink Thread Log

Livelink provides you with the option of logging the activities of the Livelink server (`llserver`), the Livelink CGI program (`Livelink`), and Enterprise Extractor process (`indexupdate`). When debug logging is enabled, Livelink writes the associated files in the *`[Livelink_install]`*`/logs` directory. By default, debug logging is disabled.

The number of `thread`*`n`*`.out` and `receiver`*`n`*`.out` files is equal to the number of threads on which the Livelink server is running. If you are diagnosing a problem, you may want to temporarily set the Livelink server to run on a single thread. The reason for this is that you can more easily find the information that you are looking for in a single file.

However, running on a single thread severely impairs the performance of the Livelink server, so make sure that you do this during periods of low usage and that you return the Livelink server to its original thread setting after you complete your diagnosis.

Method: Configure Livelink Intranet Server logging

1. On the **Livelink Administration** page, under the **Search Administration Section**, click the **Configure Debug Settings** link.
2. If prompted, type the password for the user Admin in the **Admin User Password** field, and then click **Log-in**.

Note: If you have previously logged in as the user Admin during the current Web browser session, you are not prompted to log in again at this point.

3. On the **Configure Debug Settings** page, do one of the following in the **Livelink Debug Level** list:
   - If you want to enable the creation of Level 1 logging files (`thread`*`n`*`.out`, `sockserv1.out`, and `llserver.out`), click **1**.

- If you want to enable the creation of Level 2 logging files (Level 1 + `receiver`*`n`*`.out`), click **2**.
- If you want to enable the creation of Level 11 logging files (Level 2 + `llclient`*`nnn`*`.out, llindexupdate`*`nnn`*`.out,` and `indexupdateOut`*`nnn`*`.out`), click **11**.
- If you want to disable debug logging, click **0**.
- If you want to add timing information (how long it took to perform the logged action) to the thread logs, select the **Log Livelink Timings** check box.
- If you want to add information about environment variables to the thread logs, select the **Verbose logging** check box.
- If you want to add information about the actions of system objects, including those actions relating to search queries and results, select the **Add Search and System Object Logging to Thread Logs** check box.

4. Click **Update**.
5. Restart the Livelink server on the primary Livelink host.

Method: Configure debug logging

The following table describes the available levels of debug logging, which you can select using the **Livelink Debug Level** list on the **Configure Debug Settings** page.

Note: The **Livelink Debug Level** list on the **Configure Debug Settings** page modifies the same setting as the Livelink logging list on the **Configure Server Parameters** page.

Set option levels according to this table:

Log level	Logging action
0	No logging
1	Thread logging
2	Detailed thread logging
11	Thread and CGI logging

The following table shows what log files are generated for each of the above option levels.

Level	Log Files Generated	Filename	Description
1		llserver.out	Generated each time the Livelink server restarts.
		sockserv1.out	Generated each time the Livelink server restarts.
		thread*n*.out	One file per thread is generated and continuously updated to record the data that the Livelink server sends out on its threads. The content of these files is in part controlled by thread log options.
2	Level 1 +	receiver*n*.out	One file per thread is generated to record the activities on the receiver threads.
11	Level 2 +	llclient*nnn*.out	One file is generated each time an end-user Web browser sends the Livelink CGI program a request. The number *nnn* is generated by Livelink.
		llindexupdate*nnn*.out	One file is generated each time the Enterprise Extractor starts (once per minute). The number *nnn* is generated by Livelink.
		indexupdateOut*nnn*.out	One file is generated each time the Enterprise Extractor stops (once per minute). The number *nnn* is generated by Livelink.
0		None	None of the level 1, 2, or 11 files are generated.

About the SQL Connection Log

Livelink provides you with the option of logging the communications between the Livelink server and the Livelink database. When enabled, Livelink writes database connection logs to `connect`*`n`*`.log` files in the `[Livelink_install]/logs` directory. By default, database connection logs are disabled.

The number of `connect`*`n`*`.log` files is equal to the number of threads on which the Livelink server is running. If you are diagnosing a problem, you may want to temporarily set the Livelink server to run on a single thread. The reason for this is that you can more easily find the information that you are looking for in a single file.

However, running on a single thread severely impairs the performance of the Livelink server. Make sure that you do this during periods of low usage, and that you return the Livelink server to its original thread setting after you complete your diagnosis.

Warning: Database connection logs can increase in size quickly. We recommend that you enable connection logging if it is required to diagnose a problem, but that you disable it as soon as the problem is solved.

Method: Enable or disabling database connection logs

1. On the **Livelink Administration** page, under the **Search Administration** section, click the **Configure Debug Settings** link.
2. If prompted, type the password for the user Admin in the **Admin User Password** field, and then click **Log-in**.

Note: If you have already logged in as the user **Admin** during the current Web browser session, you are not prompted to log in again at this point.

3. On the **Configure Debug Settings** page, complete one of the following steps in the Options section:
   - To enable database connection logging, select the **Log Connections** check box.
   - To disable database connection logging, clear the **Log Connections** check box.
4. The **Log Connections** check box on the **Configure Debug Settings** page modifies the same setting as the **SQL Logging** check box on the **Configure Server Parameters** page.
5. Click **Update**.
6. Restart the Livelink server on the primary Livelink host.

Livelink Admin Server Logging (OTAdmin)

You can perform the following tasks with Livelink Admin Server:

- Enable searching and indexing

Enabling Searching and Indexing

You can enable searching and indexing in one of two ways:

- By editing `opentext.ini`
- By modifying debug settings

Method: Edit opentext.ini

1. Make a backup of the `opentext.ini` file (located in the `[IStream DMS]/config` directory).
2. Open the `opentext.ini` file in any text editor.
3. Add two lines below `port=[PortAddress]` to the `[OTAdmin]` section to create the two required log files.

   For example:

   ```
   [OTAdmin]
   port=5858
   logfile=e:\opentext\Livelink800\logs\otadmin.log
   logfile=/support/livelink800/logs/otadmin.log
   loglevel=3
   ```

Modifying Debug Settings

You can modify the settings on the **Configure Debug Settings** page to change the default logging options for the Livelink Admin servers on a primary Livelink host.

By default, Livelink writes all logging information for the Livelink Admin servers on a primary Livelink host to a log file named `admserv.log`, which is stored in the logs directory of the primary Livelink installation. You can change the name and location of the Livelink Admin server log file if you want to record this information in a different file and/or location. You can also change the default logging level or modify the settings on the **Configure Debug Settings** page to specify whether you want to record the data stream between the Livelink server and the Livelink Admin server.

Method: To modify the Debug settings

1. Stop the Livelink Admin server on the primary Livelink host.
2. On the **Livelink Administration** page, click the **Configure Debug Settings** link.
3. To specify the name and/or location of the Livelink Admin server log file, select the **Log File** check box in the OTAdmin (Livelink Admin Server) section, and then type the absolute path of the file in the corresponding field.
4. To specify the log level of the Livelink Admin server log file, click one of the following values in the **Log Level** drop-down list:
   - 1, which sets **default** logging
   - 2, which sets **verbose** logging
   - 3, which sets **debug** logging

5. To log the data stream flowing from the Livelink server to the Livelink Admin server, select the **Input Log File** check box, and then type the absolute path of a log file in the corresponding field.
6. To log the data stream flowing from the Livelink Admin server to the Livelink server, select the **Output Log File** check box, and then type the absolute path of a log file in the corresponding field.
7. Click **Update**.
8. Restart the Livelink Admin server on the primary Livelink host.

Verbose Logging

Verbose logging records all arguments passed from the Web browser to the Livelink server in the thread logs. Logs generated while verbose logging is enabled can grow rapidly in size. It is recommended that you do not leave verbose logging enabled for any longer than needed to resolve any issues you might have.

Rebuilding Indexes

You might rebuild indexes for the following reasons:

- search results are not accurate (if you know what results should be returned)
- the processes that update the index get backed up to a point that the system runs out of physical disk space to work correctly
- you are migrating the LiveLink database to another location as a result of disk space problems
- you make a change like disabling full text indexing

Note: If you are having problems that you think are related to your indexes, contact Global Customer Support. first to verify that rebuilding the indexes will solve the problem.

Method: To rebuild the indexes

Step A: Delete the Indexes

1. Log in as an **Admin** user.
2. From **Livelink Administrator** (`func=admin.index`) page, in the **Search Administration** section, click **Open the System Object Volume**
3. Delete the **Enterprise Data Source** folder
4. Stop the Livelink services
5. From the **LivelinkHome\config** folder, delete the following files:
   - `otadmin.pid`
   - `otadmin.cfg`
   - `otadmin.pid.old`
   - `otadmin.cfg.old`

Warning: Do not delete `otadmin.pwd`.

6. Navigate to the directory in which you store your index
7. Delete the **Enterprise** folder (your folder may have a different name) and all the items that appear below it.
8. Log in to the database as the owner of the Livelink tables
9. Run the following statements:

```
delete from KINI where IniSection='OTIndex'
delete from KINI where IniSection='IndexObject'
```

If you are using Oracle, also run:

```
commit
```

10. Start the Livelink services.
11. From the **Livelink Administration** page, click **Reset Search Templates** and reset all search templates.
12. Click **Browse the System Volume**
13. Click the **Function** menu for the Default Admin server, then click **Resynchronize**.

Step B: Recreate the Indexes

1. From the Livelink **Administration** page (`func=admin.index`), in the **Search Administration** section, click **Open the System Object Volume**.
2. Click the **Add Item** menu and choose **Enterprise Data Source**.
3. Ensure all three directories are pointing to the location in which you want to store the index.
4. Click **Create Processes**.

   Once the processes are recreated, the database will be automatically reindexed.

Note: To test the indexes, wait a few minutes to allow the indexes to be recreated, then perform a simple search.

Viewing Operating System Logs

Often, the operating system logs contain valuable information about the status of your Tracker system.

Method: View operating systems logs (Windows 2000/2003 only)

1. From the **Control Panel**, double-click **Administrative Tools**.
2. Double-click the **Event Viewer**.

   The **Event Viewer** opens.
3. Click one of the following items to get more information about system events in that server functionality.
   - **Application Log**
   - **Security Log**
   - **System Log**

Optimizing Tracker Databases

Tracker supports the following databases:

- MS SQL Server
- Oracle

These databases can be optimized and maintained by a DBA familiar with their operation. If you have any specific questions regarding your installation of Tracker and these databases, please contact Global Customer Support.

INDEX

R

S

T

U

V

W