OpenAccess Configuration Guide

Part # OA-SDK-V560DOC-CONFIG

Preface

About this Manual

This manual describes the configuration of the OpenAccess products that include client/server and local versions of ODBC SDK, OLE DB SDK, .NET SDK, and JDBC SDK.

Specifically, the topics covered include:

- Run-time configuration of the client and server components
- Using the OpenAccess Administration tool to manage database entries
- Deploying the server
- Controlling tracing
- Tuning

Notation Convention

The courier font is used for system-dependent keywords and commands to be typed by the user or for code samples and program output. The pipe character '|' is used as a shortcut notation and separator between menus, sub-menus, and items. As an example: 'File | Open' means the Open option from the File menu.

Technical Support

Technical support is available by telephone and Email. Please provide the OpenAccess product version and the platform you are running on when calling/writing for technical support.

Telephone	**UNITED STATES:** 888-DDANSWR (888-332-6797) **Others: see** http://www.datadirect.com/support/contactus/phone/index.ssp
Email	supportlink@datadirect.com

Table of Contents

LIST OF TABLES

Chapter 1
Introduction

The configuration guide provides details related to configuring and tuning OpenAccess client and server components. Refer to this guide for details on the OpenAccess configuration settings in the OpenRDA.ini file that can be modified using the OpenAccess Administrator program or by editing the openrda.ini file.

Chapter 2
OpenAccess Configuration

This chapter details the OpenAccess configuration. Reference it to modify the configuration that was set up during the initial install of your software. Refer to this section for:

- modifying the Database Resource Directory on the server to add more data sources on the server
- modifying the Database Resource Directory on the client to connect to servers
- changing the server name, port number or the IP address
- setting disk cache parameters
- turning on trace to assist in diagnosing a problem
- understanding the settings in the OpenRDA.INI configuration file
- changing OLEDB, ODBC, JDBC, or .NET properties to customize the driver
- setting DSN-less connection

Overview

The OpenAccess product uses a Database Resource Directory (DRD) and an `openrda.ini` configuration file for configuration. The DRD contains information about OpenAccess databases. Clients interact with servers using the database names. The database names must be unique to allow a client (the OpenAccess ODBC/OLE DB/.NET/JDBC driver) to identify and contact a unique database. In the current implementation, each OpenAccess component (client and server) requires access to the DRD. This means that a DRD has to be configured for each client and for each server. In most cases, a single DRD can be copied to all clients.

A client's DRD contains the database entries for the databases it wants to access. A server only needs to have its own entries in its DRD. When an application invokes a connection request, OpenAccess client uses the database name specified by the application (or the user) to look up the IP address and the port number on which the server is running. It then uses this information to send a connection request to the server.

The `OPENRDA.INI` file contains parameters required by both the client and the server. This file must be set up properly to run any OpenAccess component. It must either be pointed to by the OPENRDA_INI environment variable or it must be located in the directory from which the OpenAccess component is run.

Configuring the OPENRDA.INI

The `openrda.ini` file contains configuration information that is used by all the OpenAccess components. Initial settings are configured during installation and can later be modified by using a text editor such as `vi` on UNIX and `notepad` on Windows. It is necessary to modify this file to turn on trace or to tune performance.

By default his file is located at {root}/config/{platform} directory and on non-Windows platforms it is pointed to by the OPENRDA_INI environment variable.

The OPENRDA.INI is divided into the following major sections:

- COMMON - configurations shared between client and the server
- CLIENT and TraceClient - configuration specific to OpenAccess ODBC/OLE DB/.NET/JDBC drivers (including local ODBC/OLE DB/.NET/JDBC)
- SERVER - configuration specific to OpenAccess Server
- ODBC32 - ODBC call tracing to be used by OpenAccess ODBC Driver or the local ODBC driver
- OLEDB - OLE DB provider tracing/configuraiton
- JDBC - JDBC driver tracing and configuration
- .NET - .NET provider tracing and configuration
- JavaIP – Java IP configuration and tracing

Each line in the file either contains a section name in [section] format or a symbol=value entry. Each symbol/value pair is under a section heading. A semicolon (;) in front of a line comments out that line. All OpenAccess processes must be shutdown before making any changes to the openrda.ini file.

Common Section

This section contains setting used by the client, server and DAM.

Table 2-1: Common Section Settings

Entry	Description	Default Value
CacheMemSize	It describes threshold for result set in terms of KB of result set size. If result set size exceeds this value then DAM starts disk caching of records. See *Disk Cache Configuration* for more details.	8192
CacheOptions	It contains parameters for controlling the location and size of the cache data file. It is made up of attribute=value pairs. All attribute names are in UPPERCASE. See *Disk Cache Configuration* for more details. Disk cache is disabled if this entry is not present or if you are not licensed for it.	
Codepage	Set the code page to display records from oledbisql. Default is to use the current system code page. This setting is available only on Windows. Use the –p option to specify codepage for ODBCISQL and OAISQL.	None
Config	The location of the platform specific configuration files	{root}/config/ {platform}
CryptoProvider	Determines the Windows cryptography provider to use. Refer to *Configuring Data Encryption* in this chapter.	None
EncryptionPackage	Encryption level for client/server configuration. 0 – No encryption (default) 1 – AES (OpenSSL) 2 – Windows (the value in CryptoProvider key controls which windows crypto package is used.	0
FETCHBLOCK_SIZE	Controls how many rows of data are fetched across the network at each time.	100
JOINBLOCK_SIZE	Controls how many rows of inner table data are retrieved in one query during Block Join Processing.	10
JoinOrderUsingSearchCondition	IP can choose to disable DAM from using the search condition for deciding the join order. To indicate PROCESSING_OFF, set the value to 0. PROCESSING_ON, set it to 1.	1
LogFileClose	Controls when the log file is closed by the tm module. 0 - log file is opened at start and then closed upon termination (default) 1 – close file after tracing each line of code. This setting is automatically changed by the OpenAccess Admin based on the trace level setting. It is set to 1 for NO TRACING and ERROR TRACING. It is set to 0 for all other options.	0
LogFileMaxSize	It supports truncation of ODBC, OLEDB, JDBC, .NET, client & server log files. User can set maximum log file size in Kilobytes. Default value is 8192. If you want to make log file size unlimited then set this value to 0. Tracing will stop after reaching the maximum limit.	8192
oa_root	Points to the root where the software has been installed	{root}
PageSize	Reserved. Size of page in bytes.	16384
ResultCacheRowBlockSize	Memory page size for the row nodes in the DAM. This number is in rows. Refer to *Configuring DAM Caching*	Defaults to FETCHBLOCK_SIZE

	for details.	
ResultCacheValBlockSize	Column values nodes. Refer to *Configuring DAM Caching.*	
SchemaPath	Points to the directory containing the schema files	{root}/schema
SchemaUpdateAllowed	Indicates if the server should allow users to change the schema information when the database uses static schema.	0
SecurityPackage	Enable use of Windows NTLM or Kerberos for authentication. It can have one of the following values: • Kerberos • NTLM • Negotiate • DBMS • Negotiate_DBMS	DBMS
SessPollWaitTime	This parameter determines the timeout of select socket call. This parameter indicates the number of seconds the socket select operation should block for. If you set it to -1 then timeout of select socket call is infinite. If you set it to 0 then select socket call is non-blocking mode. This number is time in seconds.	20
SetLocale	Set the locale for the OpenAccess client and server. The default is ".ACP". The value specified here is passed into the setlocale function with ALL as the category. If this value is not specified then the locale is set to the ANSI code page obtained from the operating system. If you don't want to modfy the local settings then set this value to "NONE". Example settings: English_USA.1252	.ACP
SQLEngineVerboseMode	Controls the level of SQL engine tracing. Applies to local driver and server. • 0X00 – No Tracing (default) • 0x01 – SQL engine tracing • 0x0F – SQL engine and IP operations tracing	0x00
SQLEngineVerboseTraceFile	Full of the file to where the SQL logging is written.	
UsePages	Reserved. 0 – enable use of pages for memory allocation 1 – disable paging	1
UseThreads	Determines whether you are running in a single threaded or multithreaded mode. The default is multithreaded if that platform supports it. ▪ 0 - disable multithreading ▪ 1 - enable multithreading (Default)	1

Client Section

This section contains settings used by the OpenAccess client and the OpenAccess Local drivers. Do not change these entries without having a good understanding of what they do.

Table 2-2: Client Section Settings

Entry	Description	Default Value
AddressSelection	Controls how a client attempts to connect when more than one ADDRESS entry is specified for a OpenAccess database entry. 0 – attempt connection to the first address and then the next at the specified port number. 1 – randomly select address and then start from first at the specified port number. 2 – attempt connection to the first address and then attempt the next one but increment the port number. 3 - Randomly select address and adjust the port number based on address entry.	0
AutoCommitOff	This option is used to control the AUTOCOMMIT mode of the client. Default AutoCommitOff is FALSE (which implies AUTOCOMMIT is ON). 0 – Start with AUTOCOMMIT on (default) 1 – Start with AUTOCOMMIT off	0
BUFFER_SIZE	This option can be used to control session buffer size. Note that the server is initializing fixed with buffer size (10240).	10240
DefaultQueryTimeout	The value to use as the query timeout when query timeout is enabled (see SupportQueryTimeout in this table) and the client application does not explicitly specify a timeout. 0 – unlimited (Default) >0 – default query timeout in seconds	0
HideQualifierOwner	Hide Qualifier and Owner from Schema information returned to client applications. 0 – disable (Default) 1 – enable. In addition to setting the option to hide the qualifier and owner, the driver information should be updated to indicate that the driver does not support qualifier and owner. This is done by updating the damipex.h. The values for SQL_MAX_OWNER_NAME_LEN (InfoNum=32) and SQL_MAX_QUALIFIER_NAME_LEN (InfoNum=34) should be changed from the default 128 to 0. Note that oainfo.ini on the client can also be configured to return the values as 0. [DAM] 32=0 34=0	0
KeyFile	Name of the client license file.	
NUM_OF_BUFFERS	This option can be used to control how many session buffers get initialized. Note that the server is initializing fixed number of buffers (10).	20
Port	This parameter can be used optionally to force the	0

Entry	Description	Default Value
	ODBC/OLE DB/.NET/JDBC driver to bind to a specific port instead of using 0 which binds it to the available port.	
PromptForUID	Support disabling of user name and password dialog box for ODBC & OLE DB client. ▪ 1 - Prompt for user name and password if all required information is not provided (Default). ▪ 0 - do not prompt for user name and password. The last user name used is passed to the IP.	1
RdaStmtOptimizeLevel	Reserved. This option is used by both client and server to decide on the following Optimize levels/settings: #define OA_OPTION_OPTIMIZE 0x0007 #define OA_OPTION_ENCRYPTION 0x0002 #define OA_OPTION_NUMERIC 0x0040 (So the lower 3 bits are for query optimization, next 3 bits are for Encryption and 7th bit is used for Numeric format.) The default value of 0x49 enables all options. - QueryOptimizeMode is 1 - The EncryptionMode value is 1 - Numeric new format is on (Note that older version client/servers use RDAStmtOptimizeLevel of 0x9 since they did not have numeric support).	0x49
RESPONSE_TIME_OUT	The time a client will wait for a response from a server. This number is time in seconds.	3600
SchemaQualifier	Prepend the specified qualifier string to schema queries. Can only be used in SQL IP mode when the SQL IP is handling the schema queries.	None
SecurityPackage	OpenAccess components running on Microsoft Windows systems can use NTLM or Kerberos for authentication. SecurityPackage can have one of the following values: • Kerberos • NTLM • DBMS (Default)	DBMS
SupportQueryTimeout	Enable/Disable the query timeout support. 0 – disable (Default) 1 – enable Enabling query timeout against the DAM disables the use of CURSOR based processing.	0

Entry	Description	Default Value
TraceFile	The location to place the trace output. Enable tracing by setting the flags under TraceClient section.	{root}/bin/ {platform}/ oaclient.log
UseSchemaLikeEscapeClause	Append ESCAPE clause to LIKE operator of schema queries. Can only be used in SQL IP mode when the SQL IP is handling the schema queries. 0 - disable (Default) 1 – enable	0

TraceClient Section

The tracing for the OpenAccess ODBC/OLE DB/.NET/JDBC and OpenAccess Local ODBC/OLE DB/.NET/JDBC is controlled by setting flags for the individual modules. Refer to the tracing section for more details on what entries can go in this section.

Table 2-3: TraceClient Section Settings

Entry	Description	Default Value
ALL	Trace settings for all modules	FATAL|SNO|ERRORS

Server Section

The server section contains setting for the OpenAccess Server. The server must be restarted before the new setting will take effect. The server only requires this section.

Table 2-4: Server Section Settings

Entry	Description	Default Value
Address	The TCP/IP address or host name of the server. This information is used by installation program and not by the OpenAccess server.	
ForkOnConnect	Determines whether you want to fork a process for each new connection or not. The default is not forking, if that platform supports multithreading. UseThreads and ForkOnConnect should not be enable at the same time. ▪ 0 – disable forking (Default) ▪ 1 - enable	0
Key	Server license key as listed on the license certificate shipped with the product.	As indicated on license certificate
Name	The name of the server. This name is used by installation program and not by the OpenAccess server.	{server}
Port	The TCP/IP port at which the server should run	1706
TraceFile	The location to place the server output. Enable tracing by setting the TraceOption entry.	{root}\bin\{platform}\oaserver.log
TraceOptions	Controls the tracing of the OpenAccess server process. Refer to the tracing section for more details. This entry is case sensitive.	all:f

ODBC32 Section

These settings are used for the OpenAccess ODBC Driver and the OpenAccess Local ODBC driver on all platforms. This ODBC level of tracing is to see what ODBC API functions are called and what their inputs and outputs are. Lower level tracing is supported by TraceClient section setting as explained in the Client section.

Table 2-5: ODBC Trace Settings

Entry	Description	Default Value
ExtendedFunctions	Bit mask to support disabling of SQLExtendedFetch, SQLMoreResults, & SQLDescribeParam ODBC API. 0 – Disable above API. 02– Enable SQLExtendedFetch 04– Enable SQLMoreResults. 08– Enable SQLDescribeParam. Note that this setting is only for Unix platform. For Windows you have to set it using ODBC DSN Configuration.	0xff
IgnoreUnicodeFunctions	Required to make the Unicode ODBC driver on UNIX work with the DataDirect driver manager. This setting controls the value that is returned when the DataDirect driver manager calls SQLGetConnectOptions(SQL_IGNORE_UNICODE_FUNCTIONS) to determine if it should make use of Unicode (W) functions exposed by the driver. 0 – use the Unicode functions (default) 1 – ignore the Unicode functions even though the driver exposes them.	0
MapDefaultWcharAsChar	Controls how the WCHAR (SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCHAR) is mapped to when binding specifies SQL_C_DEFAULT. 0 – map to SQL_C_WCHAR as per specification (default) 1 – map to SQL_C_CHAR which results in the conversion of Unicode data to MBCS. This is how Microsoft SQL Server behaves.	0
MessagePrefix	To change prefix in the error message. The industry convention is to use [Company][ODBC driver name] format.	[DataDirect][OpenAccess ODBC]
TraceFile	File to log ODBC calls to	{root}\bin\{platform}\oaodbc32.log
TraceOptions	Level of tracing to enable 0 - FATAL| SNO 1 - ERROR|Function level tracing 2 - Enable full tracing	0

OLEDB Section

These settings are used for tracing the OpenAccess OLE DB provider.

Table 2-6: OLE DB Trace Settings

Entry	Description	Default Value
IRowsetChange	IRowsetChange interface support on command object. 0 – disable (Default) 1- enable	*0*
MessageSource	It describes error message title to be displayed at the time of error condition.	*OpenAccess OLE DB*
TraceFile	File to log OLE DB calls	{root}\bin\{platform}\oaoledb.log
TraceOptions	Level of tracing to enable 0 - FATAL| SNO 1 - ERROR|Function level tracing 2 - Enable full tracing	0

JDBC Section

These settings are used for tracing the OpenAccess JDBC driver.

Table 2-7: JDBC Trace Settings

Entry	Description	Default Value
8BitChar	8 bit CHAR data present in the string. Use for ASCII string optimization. 0 – 8 bit chars not supported 1 – 8 bit chars supported (default)	1
MessagePrefix	To change prefix in the error message. The industry convetion is to use [Company][JDBC driver name] format.	[DataDirect][OpenAccess JDBC]
TraceFile	File to log JDBC calls to	{root}\bin\{platform}\oajdbc.log
TraceOptions	Level of tracing to enable 0 – FATAL|SNO 1 – ERROR|Function level tracing 2 – Enable full tracing	0
URLSubProtocol	Allows customization of the URL used to identify the OpenAccess JDBC driver. The value can be any string with no spaces: URLSubProtocol=myJdbc Now in the connection you would use: jdbc:myJDBC:memory_local	OpenAccess

.NET Section

These settings are used for the OpenAccess .NET provider. Trace settings at this level display the .NET Data Provider methods used by the application.

Table 2-8: .NET Trace Settings

Entry	Description	Default Value
TraceFile	File to log .NET calls	{root}\bin\{platform}\oanet.log
TraceOptions	Level of tracing to enable 0 – FATAL|SNO 1 – ERROR|Function level tracing 2 – Enable full tracing	0

JavaIP Section

This section contains setting used by a Java IP.

Table 2-8: JavaIp Section Settings

Entry	Description	Default Value
ByteBufferSize	Applies to the data buffer passed into the sqlipFetchIntoBuffer call. The size of ByteBuffer in bytes per row. The default is 1000 bytes. We multiply this setting with FETCH_BLOCK_SIZE to create the buffer that we pass to sqlipFetchIntoBuffer. It is up to the IP to make sure it does not exceed the buffer.	1000

CLASSPATH	Append this string value to the CLASSPATH environment variable.	Empty string
DBMSNAME	Reserved. This option is applicable when UseCatalogFunctions is enabled. Backend DBMS name such as "Microsoft SQL Server" "Oracle" "Access"	""
JVM_DLL_NAME	Name of the JVM DLL/Shared Object.	Depends on the OS. Windows- jvm.dll Solaris – libjvm.so RS6000- libjvm.a HP-UX – libjvm.sl Linux – libjvm.so
JVM_OPTIONS	Options to pass to the JVM that is loaded to run the Java IP. Each option should be separated with a "-". If a value contains a "-" then escape it with "\". Up to 32 option values can be specified. Maximus length of this string is limited to 1024 characters. Not applicable when OpenAccess is embedded in a Java based process.	Empty string
SigSegv	Allows JSQL IP to set the signal handler for the signal SIGSEGV. JSQL IP sets jsqlip_abort_handler() as the signal handler. ▪ 1 – enable (default) ▪ 0 – disable This option is applicable when mutithreading is enabled. Enabling the SigSegv handling will only terminate the server thread on which the fault occurred.	1
SQLGetTypeInfo	This option is applicable when UseCatalogFunctions is enabled. The option can be used to specify if the catalog function for returning the Data Type information should be mapped to JSQL IP function jsqlip_types(). ▪ 1 – Use jsqlip_types() catalog function (default) ▪ 0 – Do not use catalog function. JSQL IP will handle the query on the schema table OA_TYPES.	1
StringBufferSize	The size of StringBuffer in bytes per row that is passed into the IP. The default is 1000 bytes. We multiply this setting with FETCH_BLOCK_SIZE to create the buffer that we pass to sqlipFetchIntoBuffer. It is up to the IP to make sure it does not overflow the buffer.	1000
TraceJVM	Enable Tracing JVM tracing ▪ 0 – Disable (default) ▪ 1 – Enable	0
UseBasicGetColVal	For JSQL IP enables calling sqlipGetXXXColval functions to retrieve int, float, and double data. 0 – disable (default) 1 – enable This takes precedence over UseBulkFetch.	0
UseBulkFetch	For JSQL IP enables calling sqlipFetchRowsInBuffer. 0 – disabled (default) 1 – enable	0

UseCatalogFunctions	For a JSQL IP selects the use of catalog functions or schema query pass through to access the schema rowset. 1 – Use Catalog Functions (default) 0 – Use Schema Queries – the data source must set up the required schema tables.	1

NetIP Section

This section contains setting used by NET IP.

Table 2-9: NetIP Section Settings

Entry	Description	Default Value
TraceCLR	Reserved for future release. Enable Tracing of CLR ▪ 0 – Disable ▪ 1 – Enable	0

Configuring the Database Resource Directory

The Database Resource Directory (DRD) is a directory of the configured databases. The DRD is administered with the OpenAccess Administration Tool.

Each client's DRD needs to be configured with entries for every database it will connect to. Each server's DRD needs to be configured to have the databases it supports.

In Table 2-10, the flags in the required columns mean:

- Y – always needed for both client-server and local
- C/S - always needed for client-server. Not needed for local
- L – always needed for Local.
- S – always needed on the server DRD
- O - needed sometimes as specified in the description

Table 2-10: Database Attributes

Field	Description	Required
DATABASE	Name of this data source. All clients will use this name to connect this data source. Must be a single word. This is the name that will be used to refer to this database when setting up a ODBC/OLE DB/.NET/JDBC data source entry.	Y
ADDRESS	The TCP/IP address or host name of the server. You can give more than one TCP/IP address or host name (separated by a space) for redundant server configuration.	C/S
PORT	The TCP/IP port at which the server should run.	C/S
CONNECT_STRING	For C/C++ SDK: Database specific string with a maximum length of 128 characters. This string will be passed to the ipconnect function as the database name. If not specified, the Database name is used. This entry needs to be configured only on the server DRD and local ODBC/OLE DB/.NET/JDBC DRD. Client DRDs do not need this entry. For Java SDK: Name of the IP class file to load. Required for a Java IP. For .NET SDK: Name of the .NET class file that implements the .NET IP.	O
TYPE	This entry is required for the configuration of the server DRD and local ODBC/OLE DB/.NET/JDBC DRD. Client DRD does not need this entry. It determines which IP handles the access to this data source. OACSV is a system defined IP used for schema management. Any user written IPs will have associated name assigned at development time.	L, S
SCHEMA_PATH	This entry is required for the configuration of the server DRD and local ODBC/OLE DB/.NET/JDBC DRD if the IP uses the static schema feature of the DAM. OpenAccess ODBC/OLE DB/.NET/JDBC client DRD does not need this entry. The entry can be an absolute path or a path relative to the `{root}/schema` directory. Use the "." notation to specify a relative path to the schema directory (for example `./test` to specify `{root}/schema/test`).	O
REMARKS	Description of the data source	O

Using the OpenAccess Administration Tool

During the installation, the setup program installs a tool that helps to configure the OpenAccess DRD. This utility is called the 'OpenAccess Administration Tool' and is available on all platforms. It is used to configure database entries, to control tracing, and to manage the license. The Windows version has features to control many other setting through the GUI.

Running the OpenAccess Administration tool (On UNIX)

1. Execute RDAADMIN tool by running it from the command line.

   RDAADMIN supports the following command line interface:

   rdaadmin [-i *configfile***]**

RDAADMIN Command Line Options

Option	Description	Example
-i *configfile*	Configurations file to be used. RDAADMIN will use this file instead of using the file pointed to by the OPENRDA_INI environment variable.	-i openrda.ini

When you run rdaadmin, this will bring up the main menu.

Adding a Database

A Database is an entry to bind a logical name of a database to its type, to the server port and address it is running on, and to its description. Databases are the names clients use to connect to a database.

A database entry is required on a server to define a database and on a client system to allow the client to access it. If you are configuring a client's DRD to access a database on another system then you do not need to specify the TYPE or the SCHEMA_PATH fields. These are only required if you are configuring the server.

1. From the main menu, select "Add database entry".
2. Fill in each field as prompted. The lines will begin with the field name and then list the default value in brackets. To accept the default, just press Enter. To null the field, insert one space and press Enter. To customize the entry, enter your information and press enter.
3. Success or failure is indicated after all the values are entered.

Controlling Tracing

The OpenAccess Admin tool can be used to easily control tracing. It allows you to set the tracing of all the components to the following levels:

- No Tracing – turn off all tracing except for fatal errors.
- Error Tracing – errors, such as invalid query, are logged to the file.
- Major Tracing – trace only higher level events like connection, execution, etc.
- Full Tracing – trace every operation to the fullest level possible. Our support staff will ask you to set to this level and submit the generated log files.

1. From the main menu, select "Edit Trace Options".
2. Select one of the trace options
3. Select 4 to quit and go back to the main menu

You must restart the ODBC/OLE DB/.NET/JDBC driver and the server for the new trace setting to take effect.

Depending on the configuration of OpenAccess you are using, the logs generated by this setting can be found in files `{root}/bin/{platform}/oaclient.log`, `{root}/bin/{platform}/oaserver.log`, `{root}/bin/{platform}/oaodbc.log`, `{root}/bin/{platform}/oaoledb.log`, `{root}/bin/{platform}/oajdbc.log`, and `{root}/bin/{platform}/oanet.log`.

License Management

You can use the Edit License option to modify the name of the client and/or server license key files. This is required when you need to extend your evaluation or to convert from an evaluation to a permanent license.

1. From the main menu, select "Edit License".
2. Select one of the options and specify the fully qualified name of the license file. Note that the file will remain at the specified location.
3. Select 3 to quit and go back to the main menu

Running the OpenAccess Administration tool (On Windows)

Execute WRDADMIN tool by selecting the icon from the OpenAccess program group or running it from the command line. This will bring up the main window as shown in Figure 1.

Configuration File shows the file open by this tool to read the configuration. You can run this tool from command line with –i {file name} to read specific configuration file.

To accept changes you have to press **OK** before exiting from the application. To cancel all changes press **Cancel** before exiting from the application. Note that we are keeping all changes in memory until you press **OK** or **Cancel**.

Adding a Database

A Database is an entry to bind a logical name of a database to its type, to the server port and address it is running on, and to its description. Databases are the names clients use to connect to a database.

A database entry is required on a server to define a database and on a client system to allow the client to access it. If you are configuring a client's DRD to access a database on another system then you do not need to specify the TYPE or the SCHEMA_PATH fields. These are only required if you are configuring the server.

1. From the main Window, select **Database** tab. Then click on **New** button.
2. Fill in each field in **OpenAccess Database Entry** group.

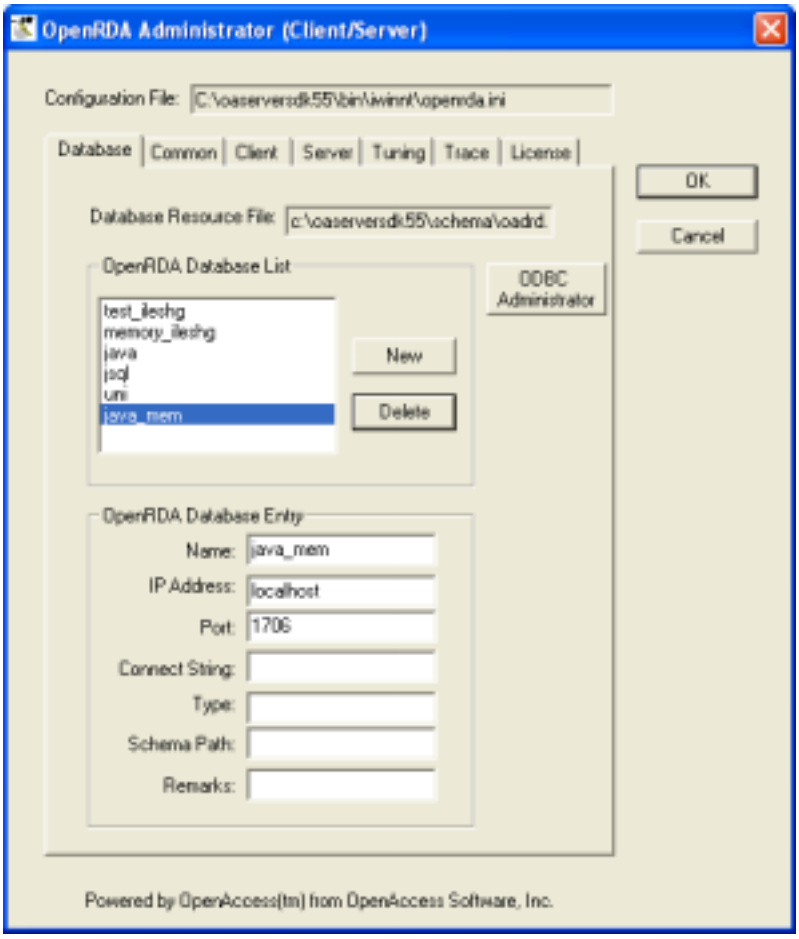

Figure 1: OpenAccess Admin for Windows

Updating a Database

1. Select Database tab
2. Select the database entry to update from **OpenAccess Database List.** This will display detail database entry in **OpenAccess Database Entry** group
3. Update field in **OpenAccess Database Entry** group

Deleting a Database

1. Select Database tab
2. Select the database entry to update from **OpenAccess Database List.** This will display detail database entry in **OpenAccess Database Entry** group
3. Now press **Delete** button

Note that **Database Resource File** shows the file open by this tool to read the database entries for all above database update operation

Controlling Tracing

The RDA Admin tool can be used to easily control tracing. It allows you to set the tracing of all the components to the following levels:

- No Tracing – turn off all tracing. Only fatal errors are logged.
- Error – Any condition treated as an error are logged.
- Major Tracing – trace only higher level events like connection, execution, etc.
- Full Tracing – trace every operation to the fullest level possible. Our support staff will ask you to set to this level and submit the generated log files.

1. Select **Trace** tab
2. Select one of the trace options
3. Edit field **File Path** shows path where to store the trace log files. Default is {root}\bin\{platform}.
4. Edit field **Max File Size** controls the maximum size of the trace file. Default is 8192Kb.

You must restart the ODBC/OLE DB/.NET/JDBC driver and the server for the new trace setting to take effect.

Depending on the configuration of OpenAccess you are using, the logs generated by this setting can be found in files **File Path**/oaclient.log, **File Path**/oaserver.log, **File Path**/oaodbc.log, **File Path**/oaoledb.log, **File Path**/oajdbc.log, and **File Path**/oanet.log.

License Management

You can modify your client and/or server license keys. This is required when you need to extend your evaluation or to convert from an evaluation to a permanent Key.

1. From the main Window, select **License** tab
2. Specify the path of the license files. These files will be referenced from the specified location therefore they should be placed at a location where they will not be deleted from.

Common Options between Client, Server and Local driver

You can modify client/server or local common options using **Common** tab.

1. From the main Window, select **Common** tab
2. If you don't want to allow user to update the schema then disable **Schema Update Allowed** option. Default is enable.
3. If you want to run your driver or server in thread safe mode instead of pure multithreaded mode then disable **Use Threads** option. Default is enable.

Client Options

You can modify client or local options using **Client** tab.

1. From the main Window, select **Client** tab
2. If you don't want to prompt for UID/PWD during SQLDriverConnect then disable **UID Prompt** option. Default is enable.
3. If you want to bind client with specific port then edit **Port** option. Default is 0.

Server Options

You can modify server options using **Server** tab.

1. From the main Window, select **Server** tab
2. If you want to run server with specific port then edit **Port** option. Default is 1706.

Tuning Options

You can tune your driver or server using **Tuning** tab.

1. From the main Window, select **Tuning** tab
2. If you want to optimize the disk cache usuage then edit parameters under **Disk Cache Options** group. See *Disk Cache Configuration* for more details.
3. **Fetch Block Size** controls how many rows of data are fetched across the network at each time. Default value is 100. You can tune this value according to your physical memory & your result row size.
4. **Response Timeout** indicates a client will wait for a response from a server. Default value is 3600 seconds.
5. **Session Poll Wait Time** indicates timeout of select socket call. Default value is 60 seconds. If you keep too low value of it then CPU will busy to poll sockets, when your server is idle. If you keep too high value of it then terminating the server will take longer time.

Enabling Tracing

The OpenAccess Server, the OpenAccess ODBC/OLE DB/.NET/JDBC clients, and the OpenAccess Local ODBC/OLE DB/.NET/JDBC drivers support extensive tracing of events to a file to allow the developer to troubleshoot and to follow the operations of the client and the server. Please note that in most cases the tracing options you can set using the OpenAccess Admin tool are sufficient – no need to manually modify files to turn on and off tracing. Tracing in OpenAccess is controlled on a module by module basis by settings in the OPENRDA.INI file (tracing for DAMISQL is controlled from the command line and is explained in *Running the DAM* chapter). All tracing is placed in the files specified by the `TraceFile` entry in the client, server, ODBC, OLEDB, JDBC and .NET sections. An existing client and server trace file is overwritten each time the process is run. The ODBC/OLE DB/.NET/JDBC trace file is not cleared by each run.

The mechanism for setting the trace flags is different for the ODBC/OLE DB/.NET/JDBC driver and the server. The server tracing is controlled by the `TraceOptions` entry in the SERVER section of the OPENRDA.INI. The ODBC/OLE DB/.NET/JDBC driver tracing is controlled by the `TraceClient`, the `ODBCxx`, the `OLEDB`, the `JDBC` and the `.NET` sections in the OPENRDA.INI. By default, the ODBC/OLE DB/.NET/JDBC driver and the server are set to only log errors. The default ODBC/OLE DB/.NET/JDBC driver tracing is `ALL=FATAL|SNO|ERRORS` for the TraceClient section and 0 for the ODBCxx, the OLEDB,the JDBC and the .NET section. The default server setting is `TraceOption= all:f`.

Event Logging when log file cannot be created or opened (On Windows only):

When the OpenAccess ODBC driver/JDBC driver/.NET Provider/OLE DB Provider cannot open the Log file for tracing, it logs an Error event in the Application Log of the Event viewer. The following information is logged in the Application Log of the Event Viewer:

Application Log

Type	Date	Time	Source	Category	Event	User	Computer
Error	6/23/2003	1:23:12 PM	OpenRDA	None	1	N/A	IBPC1

Description of the OpenAccess Error event when log file cannot be created:

The description for Event ID (1) in Source (OpenRDA) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. The following information is part of the event: Error in creating log file:c:\oasdk50\bin\iwinnt\oaodbc32.log.

Description of the OpenAccess Error event when log file cannot be opened:

The description for Event ID (1) in Source (OpenRDA) cannot be found. The local computer may not have the necessary registry information or message DLL files to display messages from a remote computer. The following information is part of the event: Unable to open log file (on Windows).

Setting OpenAccess ODBC/OLE DB/.NET/JDBC Trace Flags

The tracing for the OpenAccess ODBC/OLE DB/.NET/JDBC and OpenAccess Local ODBC/OLE DB/.NET/JDBC is controlled by entries in the TraceClient, the ODBC32 , the OLEDB, the .NET and the JDBC sections. Specifying the module name (Table 2-11) followed by the trace option sets flags for that specific module. The ALL entry sets the default tracing for all modules, except session, and can be overridden by explicit module entries. For example, adding the line RDA=MAX turns on maximum tracing on the RDA module. ODBC API calls can be traced by enabling the ODBC tracing options in the ODBC32 section. OLE DB API calls can be traced by enabling the OLEDB tracing options in the OLEDB section. .NET API calls can be traced by enabling the .NET tracing options in the .NET section. JDBC API calls can be traced by enabling the JDBC tracing options in the JDBC section.

Only the flags marked with a single or double asterisk apply to the local ODBC/OLE DB/.NET/JDBC tracing. All flags except those marked with a double asterisk apply to the client/server ODBC/OLE DB/.NET/JDBC driver tracing. For OpenAccess Local ODBC/OLE DB/.NET/JDBC, the IP level tracing is enabled by setting flags for modules SQLD and SQLS.

Table 2-11: Client Modules

Entry	Description
ALL	Trace settings for all modules
SESS	Session layer tracing
RTP	Encode/decode tracing
XM	Memory
AL	Association
RDA	RDA protocol
SUR	Sequence rule
AE	Application entity
PRESS	Presentation layer
CLI*	CLI layer
SAO	Reserved for ATI
SACF	Reserved for ATI
RDAASE	Reserved for ATI
SQLS**	Database server layer (only for local configuration)
SQLD**	Enable tracing in the IPs (only for local configuration)

Table 2-12: Client Trace Flag Settings

Flag	Description
MAX	All tracing options set
FATAL	Fatal errors
SNO	Should not happen conditions
PARM	Invalid parameters
ERRORS	General errors
MAJOR_EV	Major network event
MINOR_EV	Minor network event
INFO	Informational trace
F_TRACE	Function trace
TRIVIA	Trivial messages
TIMESTAMP	Time stamp function calls

Setting Server Trace Flags

Server trace flags work the same as for client except the notation for specifying them is different. All modules and associated trace flags are listed on a single line and the flags for each module are specified by a bit-mask. The bit-mask is specified by a hex number formed by combining the required flags from Table 2-14. The module names are listed in Table 2-13. The ALL entry sets the default tracing for each module that is overridden by explicit module entries. For example, maximum tracing on session module is turned on by adding the line :

```
TraceOptions = all:f  sess:1ff
```

The module names and the flag settings are case sensitive. The flags are the bit-mask in hex without the leading 0x. They are parsed from left to right.

Table 2-13: Server Modules

Entry	Description
all	Trace settings for all modules
sess	Session layer tracing
rtp	Encode/decode tracing
xm	Memory
al	Association
rda	RDA protocol
sur	Sequence rule
ae	Application entity
press	Presentation layer
sqls	Database server layer
sqld	Enable tracing in the IPs
sao	Reserved for ATI
sacf	Reserved for ATI
rdaase	Reserved for ATI

Table 2-14: Server Trace Flag Settings

Flag	Description	Bit Mask
MAX	All tracing options set	0x1ff
FATAL	Fatal errors	0x001
SNO	Should not happen conditions	0x002
PARM	Invalid parameters	0x004
ERRORS	General errors	0x008
MAJOR_EV	Major network event	0x010
MINOR_EV	Minor network event	0x020
INFO	Informational trace	0x040
F_TRACE	Function trace	0x080
TRIVIA	Trivial messages	0x100
TIMESTAMP	Time stamp function calls	0x200

Disk Cache Configuration

The Data Access Manager (DAM) uses disk storage for processing large result sets instead of consuming excessive memory. This note explains which queries are cached to disk, how we store the results on the disk, and the configuration parameters that control the disk cache module.

When Query Results Are Disk Cached

Results are cached to disk when they exceed the memory size as specified by the configuration parameters. We refer to this parameter as *CacheMemSize.* The disk cache feature will be enabled in the following cases:

- A query that includes post-processing options like ORDER BY, GROUP BY, DISTINCT clauses requires that DAM retrieves the entire result set and then perform the ORDER BY or GROUP BY processing. If this result set is large, then results will be disk cached and the post-processing option will be processed on disk and results will be retrieved from disk as requested by the client application. This avoids large memory requirement.
- If the IP is not cursor-based and the result set is large, DAM will disk cache results being returned by the IP. After the IP completes returning all the results, DAM will retrieve results from disk as requested by the client application.

How Query Results Are Disk Cached

Results are written to a file referred to as *CacheData* file that is created uniquely per connection. So if a user attempts to execute multiple queries that have large result sets, all these results will be stored in the same cache file. The *CacheData* file is created when we need to cache results of a query and is used for all queries on that connection. The *CacheData* file is deleted when the user disconnects. The CacheData file is created using the configuration parameters that control its initial size, the increment size and the maximum size to which it is allowed to grow.

If a result set requires to be sorted, two additional files Sort1 and Sort2 are created with the same size as the *CacheData* file which are deleted after completing the sort operation. Even if multiple statements (with large result sets) require sort operation, since the server (or Local ODBC/OLE DB/.NET/JDBC driver) can handle one statement at a time, only one sort operation will be performed at any time and only one set of sort files will be created.

Based on how many concurrent connections are expected that may require disk caching, you need to reserve disk space sufficient for these cache files.

Disk Cache File Size

To estimate the *CacheData* file size, you need to know the overhead of storing records on disk. The following overhead factors need to be used in estimating the total size:

- Overhead per record: 2 bytes, if record size is less than 65536 bytes. 6 bytes for larger size record.
- Overhead per column: 3 to 6 bytes depending upon column type & total columns in the record. The following table shows the exact overhead

Table 2-15: Overhead for each column in a record

Data Type	Total Columns in Record < 255	Total Columns in Record > 255

	255	**255**
CHAR(254), INTEGER, SMALLINT, REAL, FLOAT, DOUBLE, DATE, TIME, TIMESTAMP, NUMERIC	3 bytes	5 bytes
VARCHAR	4 bytes	6 bytes
LONGVARBINARY	6 bytes	8 bytes

Example Record Sizes

If we have 5 CHAR fields with each of 20 byte width then record area size is: 117 bytes (2 + 5 * (3 + 20)) i.e. 17% space overhead for 5 fields. It means 3.4% space overhead per field.

If we have 5 INTEGER fields then record area size is: 37 bytes. i.e. (2 + 5 * (3 + 4)) 85% space overhead for 5 fields. It means 17% space overhead per field.

Queries That Disable Disk Cache

The results of following queries will not be cached to disk even if they have large result sets:

- Query with `UNION : SELECT * FROM emp UNION SELECT * FROM dept`
- Nested Queries:

  `INSERT INTO newemp SELECT * FROM emp` – results of emp table cannot be disk cached

  `SELECT * FROM emp WHERE deptno IN (SELECT deptno FROM dept)` – Only results of EMP table can be disk cached. Not results of DEPT table.
- In JOIN queries, only the final results are disk cached, not results of the individual tables involved.
- ORDER BY is not supported on VARCHAR fields longer than 255 characters and LONGVARBINARY fields.

Configuration Parameters

The disk cache module uses the following parameters from the configuration file openrda.ini. These parameters should be set up in the Common section of the configuration file. Sample configuration file:

```
[Common]

CacheOptions=PATH=c:\cache\;INITIAL_SIZE=10;INCREMENT_SIZE=5;MAX_SIZE=50;DATABLOCK_SIZE=64

CacheMemSize=8192
```

CacheMemSize: It describes threshold for result set in terms of KB of result set size. If result set size exceeds this value then DAM starts disk caching of records.

CacheOptions: It contains parameters for controlling the location and size of the cache data file. It is made up of attribute=value pairs. All attribute names are in UPPERCASE.

- *PATH*: It describes the path to store *CacheData* and *Sort* file(s).

- *INITIAL_SIZE*: It describes initial size of file in megabytes (Mb). We recommend that you estimate this size to allow most results to be saved without having to expand it too many times. Default is 10MB.
- *INCREMENT_SIZE*: It describes next increment size of the file (in Mb). Default is 5 MB.
- *MAX_SIZE*: It describes maximum allowable size of the file (in Mb). Default is 50 MB.

DATABLOCK_SIZE: Block size of the file (in Kb). We recommend that keep this size tune with operating system file buffer size for better performance. If we increase the size of this parameter then performance becomes better. This size determines how DAM writes and read records from the cache file. Default size is 64Kb. Maximum size is 64Kb.

Configuring OLEDB Properties

The `OAINFO.INI` file contains OLEDB properties for customizing in both the local and the client/server versions. This file is optional. This file is located in {oa_root}\schema directory.

Installation will put following properties in OAINFO.INI:

[OLEDB]

DBPROP_DATASOURCENAME=test

DBPROP_DATASOURCEREADONLY=0

DBPROP_DBMSNAME=OpenAccess(R)

DBPROP_DBMSVER=05.50.0000

DBPROP_PROVIDERNAME=oaoledb

DBPROP_PROVIDERVER=05.50.20

DBPROP_MULTIPLERESULTS=0

- *DBPROP_DATASOURCENAME*: It describes the name of the data source object.
- *DBPROP_ DATASOURCEREADONLY*: It describes data source updating capability.

  0 – The data source object is updatable.

  1 – The data source object is read-only.
- *DBPROP_ DBMSNAME*: It describes the name of product accessed by the provider.
- *DBPROP_ DBMSVER*: It describes the version of product accessed by the provider.
- *DBPROP_ PROVIDERNAME*: It describes the file name of the provider.
- *DBPROP_ PROVIDERVER*: It describes the version of the provider.
- *DBPROP_MULTIPLERESULTS*: It enable/disable IMultipleResults support.

  0 – Disable

  1 – Enable

DSN-less Connection

All the required information to connect to a data source can be supplied in the connect string. This allows the ODBC driver to be used to connect to a data source without the need to perform any ODBC related or OpenAccess related configurations on the user's system. Note that you can't use comma (,) in your custom connect string. We use comma for our internal purpose.

Local ODBC Driver:

Driver={DataDirect OpenAccess
Local};Database=test_local;TYPE=OACSV;CONNECT_STRING=;

SCHEMA_PATH=.\test;

Client/Server ODBC Driver:

Driver={DataDirect OpenAccess};Database=test_ileshg;TYPE=OACSV;CONNECT_STRING=;

SCHEMA_PATH=.\test; PORT=1706;ADDRESS=208.36.7.144;

Client/Server OLE DB Provider:

Database=test_{server_name};PORT=1706;ADDRESS=208.36.7.144;TYPE=OACSV;CONNECT_STRING=;SCHEMA_PATH=.\test;

Database=memory_{server_name};PORT=1706;ADDRESS=208.36.7.144;TYPE=MEM;CONNECT_STRING=memory;SCHEMA_PATH=;

Local OLE DB Provider:

Database=test_local;TYPE=OACSV;CONNECT_STRING=;SCHEMA_PATH=.\test;

Database=memory_local;TYPE=MEM;CONNECT_STRING=memory;SCHEMA_PATH=;

ADO connection using Local OpenAccess ODBC Driver and Microsoft OLE DB to ODBC bridge:

connect = "Provider=MSDASQL;Driver={DataDirect OpenAccess
Local};Database=memory_local;user id = pooh;
password=bear;TYPE=MEM;CONNECT_STRING=memory;SCHEMA_PATH=;"

ADO connection using Local OpenAccess OLE DB Provider

connect = "Provider=LocalOpenAccess;Database=memory_local;user id = pooh;
password=bear;TYPE=MEM;CONNECT_STRING=memory;SCHEMA_PATH=;"

Load Balancing / Redundant Servers

Enhancements in 5.5 allow a client to be configured to connect to a randomly selected server from a list of configured servers. Please refer to the Programmer's Guide for details of operation.

Configurations supported:

- Client will sequentially attempt to connect to a server until it succeeds or runs out of servers to connect to. The servers can be on same machine or different machines. A server can accept a connection but redirect the client to the next server.

- Client can randomly select the server to attempt connection to. The servers can be on same machine or different machines.

Run Multiple Instances on the Same Machine

Server Configuration

1) Start each instance on a different port number using the -p option to the server. If the first one is started on port 1706, then the rest must be started on sequential port numbers, 1707, 1708, etc.

oaserver -p 1706

Client Configuration

1) Modify the Openrda.ini on the client side to add the following entry in the [CLIENT] section.

[CLIENT]

AddressSelection=3

This tells the client to randomly select the address from the IP Address list and to adjust the port number.

2) Run OpenAccess Administration tool and edit the database entry to modify the IP Address field to duplicate the host name to match the number of server instances. Assuming the value is currently set to HOST1 and you have two instances of servers running on HOST1, then you would add two entries for HOST1 separated by space:

IP Address: HOST1 HOST1

Leave the port number to value that the first instance is running on.

Now run your tests and monitor that on the server both processes are getting hit.

Run Multiple Instances on Different Machines

Server Configuration

1) Start each instance of the server on the configured port number (say 1706). No special configuration is required.

Client Configuration

1) Modify the Openrda.ini on the client side to add the following entry in the [CLIENT] section.

[CLIENT]

AddressSelection=1

This tells the client to randomly select the port address from the IP Address list. If you want the connections to the servers to be attempted in the order listed then set the AddressSelection property to 0 (which is the default).

2) Run OpenAccess Administration tool and edit the database entry to modify the IP Address field to enter the host names to match the number of server instances. Assuming the value is currently set to HOST1 and you have two instances of servers running on HOST1 and HOST2, then you would append the HOST2 entry:

IP Address: HOST1 HOST2

Leave the port number to value that corresponds to the port number the servers are running on.

Now run your tests and monitor that on the server both processes are getting hit.

Configuring Row Caching in the DAM

Each query being processed maintains a cache of row and value nodes to handle the query results. The Row nodes are allocated as one large memory block and memory is split into row node and added to the row cache. RowBlockSize is the number of ROW nodes allocated as one large memory block. Similarly ValBlockSize is the number of VALUE nodes allocated as one large memory block. This allows fewer memory allocations to be done to process query results.

When dam_allocRow is called, we use a row from the cache of row nodes. When dam_freeRow is called, the row is returned back to the cache. Row is also returned back to the cache when results are returned to client application. If the cache becomes empty, we allocate another memory block for RowBlockSize row nodes. So typically in cursor mode with no post-processing, even a large query will get processed using just one memory allocation for RowBlockSize row nodes. But if we expect application to use large FetchBlockSize and want to avoid allocating large memory blocks, we can keep the RowBlockSize smaller so that multiple blocks will be allocated to handle a FetchBlockSize records. Value node cache is used when IP calls dam_addValToRow to provide column values. When row is freed, all value nodes of that row are returned back to value node cache. Once query is fully processed, the entire cache of row and value nodes is freed.

DAM in default configuration sets the RowBlockSize to be same as FetchBlockSize and ValBlockSize to be RowBlockSize*10. If IP wants to configure these sizes, it can configure the RowBlockSize and ValBlockSize in the openrda.ini.

[Common]

ResultCacheRowBlockSize=50

ResultCacheValBlockSize=100

Configuring Data Encryption

All traffic between a OpenAccess client and a OpenAccess server can be secured using Windows or OpenSSL encryption. The flexibility to choose a cross platform package like OpenSSL or Windows cryptography allows encryption to be deployed in a Windows only solution or in mixed platform solutions. OpenSSL provided the additional benefit of avoiding restrictions in exporting software that uses encryption.

If OpenSSL encryption package is selected then we expect the required functions to be supplied in libeasy32.dll (libcrypt.so on Unix). These are the default file names for OpenSSL. You can implement your own libraries based on some other algorithms as long as you comply with the OpenSSL interface used by OpenAccess.

For Windows only environment you can configure OpenAccess to use your system's default Windows Cryptographic services or specify a specific Windows Cryptographic Service provider by setting the CryptProvider key in the OpenrRDA.ini file.

On Windows the OpenAccess Administration tool controls encryption mode. Click on the COMMON tab to select the encryption mode. If you select AES (OpenSSL) then you must make sure the OpenSSL DLL libeay32.dll is in {oa_root}\bin\iwinnt folder of the client and SDK installs. The server must be restarted after changing the encryption mode.

You can get the OpenSSL DLL from our website from the same page you downloaded the SDK from. Full OpenSSL install can be downloaded and installed from http://www.slproweb.com/products/Win32OpenSSL.html

On Unix the encryption mode is controlled by editing the openrda.ini file (accessible through the OPENRDA_INI environment variable) to modify the EncryptionPackage key under the [COMMON] section. Value of 0 is no encryption (the default). A value of 1 is OpenSSL. The entry below enables OpenSSL.

```
[COMMON]

EncryptionPackage=1
```

On Unix you must have a OpenSSL library libcrypto.so installed in /usr/lib or in a directory that is in LD_LIBRARY_PATH. We have tested with OpenSSL versions 0.9.7c on windows and 0.9.7a on Linux.

Configuring Windows Cryptographic Service Provider

The CryptoProvider key in the [COMMON] section of OpenRDA.ini can be used to specify which Microsoft Cryptographic Service provider to use. If the key is not specified then the default cryptographic Service Provider will be used. The same Cryptographic Service Provider (CSP) must be selected on all systems from which the OpenAccess components will communicate with each other.

Below is a list obtained from the Microsoft's MSDN web site. The CSP that is available on a system depends on the version of Windows and the country it is running in. Windows XP2 in the USA and Canada supports Microsoft Strong Cryptographic Provider as the default. Windows 2000 Professional appears to support the "Microsoft Base Cryptographic Provider v1.0".

```
Microsoft Base Cryptographic Provider v1.0
Microsoft Enhanced Cryptographic Provider
Microsoft Strong Cryptographic Provider
Microsoft RSA Signature Cryptographic Provider
Microsoft RSA SChannel Cryptographic Provider
Microsoft Base DSS Cryptographic Provider
```

```
Microsoft Base DSS and Diffie-Hellman Cryptographic Provider
Microsoft Enhanced DSS and Diffie-Hellman Cryptographic Provider
Microsoft DH SChannel Cryptographic Provider
Microsoft Base Smart Card Cryptographic Provider
```

OpenAccess only supports the use of CSPs that support PROV_RSA_FULL and CALG_RC4 stream based algorithm.