Using emissary RT: Servers

Overview

Emissary RT: Servers is an ODBC driver designed to allow network admins and developers to query and update Windows Server services, including DHCP, DNS, and the event log, from an existing or custom made ODBC-enabled application. Service information (such as DHCP scopes and leases, DNS records, Windows events, etc) is presented to the application as rows across a collection of related tables, and most standard SQL commands can be used to read and/or write this data.

By making use of emissary RT's lightweight, yet powerful Real-Time SQL engine, the latest service information can be shown, or static snapshots for increased performance. Updates take place immediately and allow for powerful manipulation of records not provided by the normal MMC tools. No need for interim or temporary tables, programming code, or synchronization to accomplish SQL controlled interoperability with Windows Server services. With the proper credentials, *emissary RT: Servers* can also connect to remote servers, allowing for distributed functionality, perfect for a web application or client application environment.

Administration Usage Examples

Examples can be performed via an ODBC-enabled application such as MS Access or Excel, using the GUI or SQL commands. No programming required.

- Systematically create or rename DNS A records in a specific domain, generating the name using related info, such as IP address
- Create boilerplate DNS TXT records for all domains for legal notifications or technical configurations
- Create DNS CNAME records for all DNS A records that match specified criteria
- Create DHCP reservations for all DNS A records in a specific domain (or vice versa)
- Delete DHCP leases or reservations matching a certain age range
- Adjust start and end IP across multiple DHCP scopes
- Search for a combination of keywords in the Windows event log during a specified time period

Programming Usage Examples

Examples can be performed in your custom web or desktop application via an ODBC connection.

- Link DHCP and DNS record information to foreign tables in a Document Management System

- Periodically query DHCP leases for a list of active network devices in applications that deploy batch installs/updates
- Synchronize DNS record information with a centralized network control application

Shown are only some of the possibilities. Countless DHCP, DNS, and event log tasks can be made easier via leveraging the power and flexibility of ODBC and SQL queries.

Available Types

File information is presented to the application as rows across a collection of related tables. These tables are as follows (follow links for full field listings and additional usage notes):

Table	Description
DHCPScope	DHCP Scope information, such as network address and mask, name, IP range, etc
DHCPLease	DHCP Lease information, such as parent scope, IP, hostname, expiration, reservation, etc
DNSZone	DNS Zone information, such as root domain name, zone type, scavaging frequency, etc
DNSDomain	DNS (sub)domain information, such as parent zone and domain name
DNSRecord	DNS record information, such as parent domain, Name, Type, Data, TTL, etc
Event	Windows event log information, such as generation date, source, message, type, etc
Settings	Current data source configuration options, allowing run-time reconfiguration
Errors	Log containing errors generated from any query operations

Table - DHCPScope

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
IP	Char	Yes*	Network address of scope. *Must be specified for INSERT statements
Mask	Char	Yes*	Network mask of scope. *Must be specified for INSERT statements
Name	Char	No	Name
Comment	Char	No	Comment/Description
Host	Char	Yes	DHCP server where scope resides
State	Integer	No	0 = Scope active, 1 = Scope disabled
Start	Char	No	Starting IP address of scope's range
End	Char	No	Ending IP address of scope's range
Duration	Integer	No	Default duration of leases (in seconds)

Usage Notes

1. INSERT and DELETE statements run against the 'DHCPScope' table will create and delete DHCP scopes on the target server, respectively
2. As noted, the 'IP' and 'Mask' fields must be specified, at minimum, for INSERT statements. Any non read-only field may also be specified in INSERT statements to be applied to newly created scopes
3. DELETE statements run against the 'DHCPScope' table will delete corresponding entries in the 'DHCPLease' table

Table - DHCPLease

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
ScopeID	Integer	Yes	Foreign Key of parent scope
IP	Char	Yes*	IP address of lease. *Must be specified for INSERT statements
Hardware	Char	No	Hardware (MAC) address of lease in hexadecimal. Must be specified for INSERT statements
Name	Char	No	Name. Must be specified for INSERT statements
Comment	Char	No	Comment/Description
Expiration	Timestamp	No	Expiration date of lease. NULL indicates inactive reservation. 0000-00-0000 00:00:00 indicates active reservation/infinite lease
Owner	Char	Yes	DHCP server where lease resides
Type	Char	Yes	Lease type ("None", "DHCP", "BOOTP", etc)
Reservation	Integer	No	Lease is a reservation (0 = No, 1 = Yes). UPDATE statements changing 1 to 0 will cause the lease to be deleted

Usage Notes

1. INSERT and DELETE statements run against the 'DHCPLease' table will create and delete DHCP leases on the target server, respectively
2. The appropriate scope (address and mask) must exist to create a lease
3. As noted, the 'IP', 'Hardware', 'Name' fields must be specified, at minimum, for INSERT statements. Any non read-only field may also be specified in INSERT statements to be applied to newly created leases

Table - DNSZone

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
Name	Char	Yes*	Domain name / Network Address. *Must be specified for INSERT statements
Reverse	Integer	Yes	Reverse Lookup Zone (0 = No, 1 = Yes)
Type	Integer	Yes	Zone type (1 = Primary, 2 = Secondary, 3 = Stub)
DSIntegrated	Integer	Yes*	Zone is Active Directory integrated (0 = No, 1 = Yes). *Must be specified for INSERT statements
Scavenge	Integer	Yes	Stale resource record scavenging active (0 = No, 1 = Yes)
ScavengeNoRefresh	Integer	No	Scavenging no-refresh interval (in seconds)
ScavengeRefresh	Integer	No	Scavenging refresh interval (in seconds)
FileName	Char	Yes	Zone file name
Master	Char	Yes	Master server (for secondary and stub type zones. NULL for primary zones)
Paused	Integer	No	Zone is paused (0 = No, 1 = Yes)

Usage Notes

1. INSERT and DELETE statements run against the 'DNSZone' table will create and delete DNS zones on the target server, respectively
2. As noted, the 'Name' and 'DSIntegrated' fields must be specified, at minimum, for INSERT statements. Any non read-only field may also be specified in INSERT statements to be applied to newly created zones
3. To INSERT a reverse lookup zone, name should be formatted in reverse format with "in-addr.arpa" TLD suffix. *E.g.* "0.168.192.in-addr.arpa" for the 192.168.0/24 network
4. INSERT statements run against the 'DNSZone' table will create the corresponding domain entry in the 'DNSDomain' table
5. DELETE statements run against the 'DNSZone' table will delete corresponding entries in the 'DNSDomain' and 'DNSRecord' tables

Table - DNSDomain

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
ZoneID	Integer	Yes	Foreign Key of parent zone
Name	Char	Yes	Name

Usage Notes

1. 'DNSDomain' is a read-only table. No UPDATE, INSERT, or DELETE statements may be run against it
2. INSERT statements run against the 'DNSZone' table will create a corresponding entry in the 'DNSDomain' table

Table - DNSRecord

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
DomainID	Integer	Yes*	Foreign Key of parent domain. *Must be specified for INSERT statements
Type	Char	Yes*	Record type ("A", "NS", "CNAME", "TXT", etc). *Must be specified for INSERT statements
Name	Char	Yes*	Name. NULL when record applies to domain itself. *Must be specified for INSERT statements
FQDN	Char	Yes	Fully qualified record name
Data	Char	No	Applicable data for record type (IP Address, domain name, text, etc). Space delimited for records requiring multiple values. Use quotes for embedding spaces (TXT records). Must be specified for INSERT statements
TTL	Integer	No	Time-to-live value of the record

Usage Notes

1. INSERT and DELETE statements run against the 'DNSRecord' table will create and delete DNS records on the target server, respectively
2. As noted, the 'DomainID', 'Type', 'Name', and 'Data' fields must be specified, at minimum, for INSERT statements. Any non read-only field may also be specified in INSERT statements to be applied to newly created records
3. The appropriate DomainID must exist to create record
4. The 'data' field must be formatted as records appear in the zone file. *E.g.* "192.168.0.100" for an "A" record, "10 mail.example.com" for an "MX" record, "example.com." for a "CNAME" record, etc

Table - Event

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
Log	Char	Yes	Event log containing event ("Application", "Security", "System", etc)
GeneratedDate	Timestamp	Yes	Date/time event was created
Source	Char	Yes	Application or system that generated event
Level	Char	Yes	Severity Level ("Information", "Warning", "Error", etc)
Message	Char	Yes	Full description of the event
Record	Integer	Yes	Event record number
EventID	Integer	Yes	Event ID
Computer	Char	Yes	Name of the computer generating the event

Usage Notes

1. 'Event' is a read-only table. No UPDATE, INSERT, or DELETE statements may be run against it

Table - Settings

Field	Type	Read-only	Description
Key	Char	Yes	Data source setting keyword. See Settings and Options
Setting	Char	No	Data source setting value. See notes
Description	Char	Yes	Full description of data source setting

Usage Notes

1. INSERT and DELETE statements may not be run against the 'Settings' table.
2. Initial values of the 'Setting' field will reflect data source settings as configured from the ODBC manager.
3. Changes made to 'Setting' field will immediately update data source settings for the duration of the connection (settings will revert to permanent values upon disconnect). Changes to DHCP, DNS, Events, Server, or Logs will cause *emissary RT: Servers* to rebuild all caches.

Table - Errors

Field	Type	Read-only	Description
ID	Integer	Yes	Primary Key (unique ID) for row
Date	Timestamp	Yes	Date/time the error occurred
Error	Char	Yes	Error description and details
Query	Char	Yes	SQL query that caused the error

Usage Notes

1. INSERT and DELETE statements may not be run against the 'Errors' table.
2. The 'Errors' table is automatically purged prior to executing UPDATE, INSERT, or DELETE statements. The table should be checked for error details immediately after an unsuccessful query fails to execute.

Supported SQL Syntax

SELECT Statement

```
SELECT select_expression [, select_expression . . .]
  [FROM table_expression
    [WHERE general_expression]
    [ORDER BY general_expression [ASC | DESC], . . .]
    [LIMIT [row_offset,] row_count ] ]
```

Note: SELECT statements used with a FROM command will retrieve data from the table(s) specified in the table_expression (see below). Usage without a FROM command will return a single row, executing any specified expressions in the select_expression (see below). At least one select_expression is required.

UPDATE Statement

```
UPDATE table_expression
  SET column1_name=general_expression [, column2_name=general_expression . . .]
  [WHERE where_expression]
  [ORDER BY order_expression [ASC | DESC], . . .]
  [LIMIT [row_offset,] row_count ]
```

Note: UPDATE statements used with an ORDER BY command will control the order in which file operations are performed. This can be useful if the order of updating filenames may otherwise cause a name collision with pre-existing files. If the SET command includes any expressions with column names, the value of the field in the currently updating row will be used. LIMIT will constraint which files are updated from the total UPDATE rowset.

INSERT Statement

```
INSERT INTO table_name
  [(column1_name, ...)]
  {VALUES | VALUE} (general_expression, . . .)
```

```
INSERT INTO table_name
  SET column1_name=general_expression [, column2_name=general_expression . . .]
```

Note: INSERT statements may use either syntax shown above. If the first syntax is used without specifying column names, the number of VALUES/VALUE expressions specified must equal the number of columns in the table. For read only fields, the value specified is ignored.

DELETE Statement

```
DELETE {table_name[.*] | *}
  FROM table_expression
```

```
    [WHERE general_expression]
    [ORDER BY general_expression [ASC | DESC], . . .]
    [LIMIT [row_offset,] row_count ]
```

Note: DELETE statements used with an ORDER BY command will control the order in which file operations are performed. LIMIT will constraint which files are deleted from the total DELETE rowset.

Select Expressions

```
{general_expression | [table_name.] { * | column_name} } [[AS] alias]
```

Note: Table and column names may be delineated using the ` character.

Table Expressions

```
{table1_name} [[AS] alias] [, {table2_name} [ [AS] alias] . . .]
  [[INNER | LEFT [OUTER] | CROSS] JOIN table_name
    [ON general_expression] . . .]
```

Note: Table and column names may be delineated using the ` character. Comma separated tables specified after the first table in a table_expression before JOIN commands will be treated as CROSS JOINed tables.

General Expression Operators and Functions

Literal	Operands/Arguments	Precedence	Description	
=	binary	1	Assign. Recognized in UPDATE and INSERT statements	
=	binary	7	Equal. Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)	
		binary	7	Not equal. Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
>	binary	7	Greater than. Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)	
			Greater than or equal. Case	

>=	binary	7	insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
<	binary	7	Less than. Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
<=	binary	7	Less than or equal. Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
LIKE	binary	7	String comparison with wildcard matching. '%' matches 0 or more characters. '_' matches 1 character. Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
+	binary	11	Add. Parses strings to numeric equivalent.
-	binary	11	Subtract. Parses strings to numeric equivalent.
*	binary	12	Multiply. Parses strings to numeric equivalent.
/	binary	12	Division. Parses strings to numeric equivalent.
%	binary	12	Modulo. Parses strings to numeric equivalent.
IS	binary	7	Equal (NULL safe). Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
IS NOT	binary	7	Not equal (NULL safe). Case insensitive for strings, case sensitive for BINARY type. Returns boolean value (0 = false, 1 = true)
AND	binary	2	Logical AND. Returns boolean value (0 = false, 1 = true)
OR	binary	4	Logical OR. Returns boolean value (0 = false, 1 = true)

NOT	unary	5	Logical NOT. Returns boolean value (0 = false, 1 = true)
CONCAT	Variable	Function	String concatenation. CONCAT(string1, ...)
CONVERT	2	Function	Type conversion. CONVERT(value, type)
LOCATE	2/3	Function	Return starting position of substring. LOCATE(substring, full string, [start index])
SUBSTR	2/3	Function	Return substring. SUBSTR(string, [start index,] num of chars)

SQL Types

CHAR, VARCHAR, LONG VARCHAR, BINARY, SMALLINT, INTEGER, FLOAT, DOUBLE, TIMESTAMP

Miscellaneous ODBC Support

"{d '1995-01-15'}" style date literals, unnamed parameters via '?' literals, single prepare/multiple execution model with parameter updating, thread-safety. Contact Synthetic Dreams regarding any further ODBC support questions.

Performance and Considerations

To increase efficiency of processing SQL queries against Windows Server services, *emissary RT: Servers* makes use of an in-memory caching system. This cache (if enabled in the Data Source options) creates a snapshot of the respective service, greatly increasing performance. This cache is initially built when the ODBC connection is established, and is maintained for the lifetime of the connection. If the snapshot is disabled, *emissary RT: Servers* will rescan and update its cache (if necessary) when it executes a SQL query. This ensures all data is 100% up-to-date.

When making use of *emissary RT: Servers* in custom applications, as the cache is built at connection time and is maintained for the lifetime of the connection, it is important to reuse the ODBC connection when possible. This can be more challenging in a web application environment, and may require changes to both the web server configuration and API used. An example includes PHP's odbc_pconnect function and a compatible Apache configuration (non-CGI mode), which creates a persistent connection across each request (for the session lifetime).

When executing queries containing related tables (either via JOIN or appropriate WHERE clauses), *emissary RT: Servers* is optimized for predicates comparing the equality of foreign keys to primary keys. *E.g.* "SELECT *FROM DHCPScope LEFT JOIN DHCPLease ON DHCPScope.ID = DHCPLease.ScopeID", "SELECT* FROM DHCPScope, DHCPLease WHERE DHCPScope.ID = DHCPLease.ScopeID", "SELECT * FROM DNSZone JOIN DNSDomain ON DNSZone.ID = DNSDomain.ZoneID JOIN DNSRecord ON DNSDomain.ID = DNSRecord.DomainID", *etc*. Predicates may contain additional expressions, as long as OR operators do not allow for potential additional matches in the join. Any non-optimized predicate with valid syntax may be used, but performance will degrade significantly, as the system must internally perform a full cross join.

Additionally, all tables are indexed against their ID column, and will perform significantly faster with WHERE clauses that select for specific IDs, via inline values and/or parameters. As with related table optimization above, WHERE clauses optimized for ID indices may contain additional expressions, as long as OR operators do not allow for potenital additional matches.

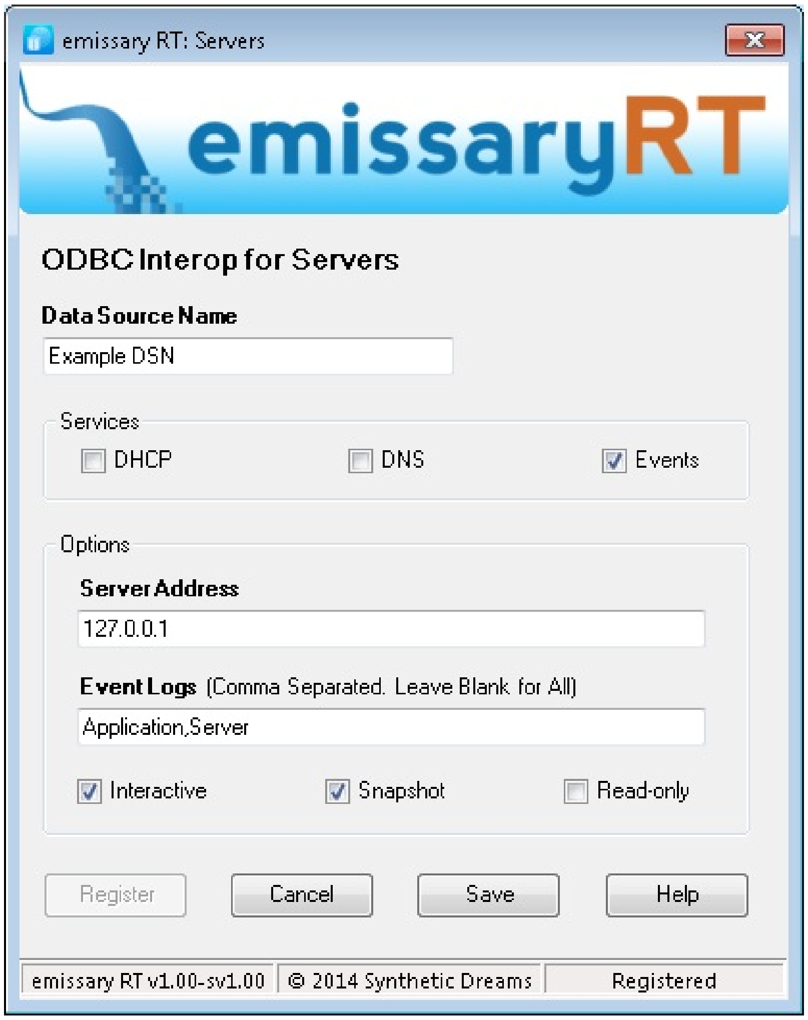
The execution time involved for a SQL query is dependent on the number of records in the service being queried, the speed of the underlying hardware hosting and querying the services, how many services are being queried, *etc*. Because it may be desired to execute a query that may take a significant time to process, *emissary RT: Servers* provides an interactive mode that shows both a progress meter, and allows the cancelation of a query. Note - canceling an INSERT, UPDATE or DELETE query is not ACID compliant - changes are made to a service in real-time, and are not automatically rolled back. Please construct a corresponding SELECT query for testing before executing any potentially destructive INSERT, UPDATE or DELETE queries.

Configuring a Data Source

Before making use of *emissary RT: Servers*, it is necessary to configure one or more desired data sources. A data source indicates which (one or more) Windows Server services *emissary RT: Servers* will query, the target machine, if the access is read-only, *etc*. A full list of all options and their effect can be found in the Settings and Options guide.

DSNs can be created, configured, or deleted from the Microsoft ODBC Manager, typically found in the Administrative Tools menu. It is also important to use the correct version of the ODBC Manager depending on the architecture (32/64bit) of the ODBC-enabled application. Please refer to Microsoft's documention for further details.

Minimally, an *emissary RT: Servers* data source must be configured with services and server address defined, as shown below (with the default options enabled).



ODBC Interop for Servers

Data Source Name: Example DSN

Services: [ ] DHCP [ ] DNS [✓ Events]

Options

Server Address: 127.0.0.1

Event Logs (Comma Separated. Leave Blank for All): Application,Server

[✓ Interactive] [✓ Snapshot] [ ] Read-only

emissary RT v1.00-sv1.00 | © 2014 Synthetic Dreams | Registered

Settings and Options

The following attributes may be configured for each data source (and may be reconfigured at run-time):

Option	Settings Key	Description
DHCP	DHCP	"True" = Query the DHCP service on the target server, and enable the "DHCPScope" and "DHCPLease" tables. "False" = Do not query the DHCP service on the target server, and disable the "DHCPScope" and "DHCPLease" tables
DNS	DNS	"True" = Query the DNS service on the target server, and enable the "DNSZone", "DNSDomain", and "DNSRecord" tables. "False" = Do not query the DNS service on the target server, and disable the "DNSZone", "DNSDomain", and "DNSRecord" tables
Event	Event	"True" = Query the event log on the target server, and enable the "Event" table. "False" = Do not query the event log on the target server, and disable the "Event" table
Server Address	Server	The address of the server to query. May be IP or name, including localhost
Event Logs	Logs	Comma delimited list of event logs to query on the target server (such as "Application", "System", "Security", etc). Leaving this field blank will query all logs found on the target system
Snapshot Mode	Snapshot	"True" = The cache will only be populated with service data at connection time, increasing performance. "False" = The cache will be updated at query time, ensuring 100% up-to-date information
Interactive Mode	Interactive	"True" = When updating the cache, or executing a INSERT, UPDATE, or DELETE query, a progress meter will be shown to the user. The dialog will allow users to cancel queries. "False" = no dialog boxes will be displayed to the user. "False" must be used when making use of *emissary RT: Servers* in custom applications that cannot interact with the desktop, such as PHP or ASP.NET web apps.
Read-only Access	ReadOnly	"True" = INSERT, UPDATE, and DELETE queries are disabled. "False" = INSERT, UPDATE, and DELETE queries are enabled.

Reconfiguration at Run-Time

In addition to specifying configuration options for an *emissary RT: Servers* data source, these settings may also be changed during run-time if desired. Any run-time changes made will immediately affect the ODBC connection, but are temporary for that connection only. Future connections will use the settings as defined by the DSN.

To reconfigure a data source at run-time, the "Settings" table may be queried and updated. Each row of the table contains a Key and Value corresponding to a data source configuration option. A full list of all options and their effect can be found in the Settings and Options guide. Note: changing the DHCP, DNS, Events, Server, or Logs options may cause a rebuild of the internal cache.

Registering emissary RT: Servers

For trialing purposes, *emissary RT: Servers* may be freely downloaded and used. When unregistered, the system is fully functional with the exception of imposing a limit of returning and/or affecting 50 records for any query. Once purchased and registered, this restriction is lifted.

Registering *emissary RT: Servers* may be performed from any DSN configuration dialog, using the "Register" button. The system will prompt for the license key received when purchasing the product, and can register automatically via the Internet, or manually via email by following the provided instructions. Manual registration may be performed on a different machine than where *emissary RT: Servers* is installed.