EventReporter Configuration Documentation

version 19.4

Adiscon GmbH

September 03, 2026

Contents

Manual **1**

Getting Started 1

Installation 1

System Requirements 2

Understand the Components 3

Collect Windows Events 3

Creating an Initial Configuration 4

Process and Filter 5

Store and Forward 6

Operate and Troubleshoot 6

Tutorials 7

Tutorial: Prepare EventReporter Data for Adiscon LogAnalyzer 7

Tutorial: Collect Windows Events and Write Them to a File 8

Tutorial: Forward Windows Events to a Syslog Server 9

Tutorial: Forward Windows Events via TLS to rsyslog 9

Tutorial: Forward Windows Events via SETP 11

Tutorial: Send Matching Windows Events by Email 12

Tutorial: Write Windows Events to Microsoft SQL Server 13

Tutorial: Integrate EventReporter with a Custom Database Schema 15

Tutorial: Export the Configuration and Create a Debug Log 17

Configuration 18

Configuring EventReporter 18

Client Options 20

Client Tools 22

Using File based configuration 27

General Options 30

License 31

General 33

Debug 36

Engine 38

Queue Manager 43

Core concepts 44

EventReporter services 45

Information Units 46

Rules 47

The Rule Engine 50

EventReporter filter conditions 58

Actions 59

Services 60

Heartbeat 61

MonitorWare Echo Reply 63
Event Log Monitor V1 64
Event Log Monitor V2 74
Filter Conditions 82
Global Conditions 84
Date Conditions 85
Operators 86
Filters 87
General 88
Date/Time 90
InformationUnit Type 92
Event Log Monitor 94
Event Log Monitor V2 96
Custom Property 98
Extended Number Property 99
Extended IP Property 100
File Exists 102
Store Filter Results 103
Actions 103
ODBC Database Options 104
OLEDB Database Action 111
File Logging Options 117
Event Log Options 127
Send Email 129
Net Send 134
Send SETP 135
Syslog Forwarding 139
Send DTLS 152
Call RuleSet 154
Compute Status Variable 155
Discard 156
Resolve Hostname Action 157
Set Property 158
Set Status 159
Play Sound 160
Start Program 161
Multiple RuleSets - Rules - Actions 162
FAQ 163
Why were dynamic log files sometimes not rotated in older EventReporter versions? 163
Why were dynamic log files sometimes not rotated in older EventReporter versions? 164
Log Rotation Naming Convention Change in EventReporter 19.x 164
Why does log rotation fail when using ZIP compression in EventReporter? 166

Are EventReporter products affected by recent OpenSSL CVEs? 167
Troubleshooting the Start Program action in EventReporter 168
Queue Buildup During SQL Server Table Cleanup Operations in EventReporter 170
Recommended Service Stop Order for EventReporter Maintenance 173
Running EventReporter on a Windows Cluster Server 174
How to Export EventReporter Settings for a Support Call 175
Why do log files remain locked when multiple rules write to the same file? 176
Is MariaDB supported by the ODBC action? 178
How Do I Perform a Repeatable Deployment of EventReporter? 179
How to Copy the EventReporter Configuration to Other Servers 180
How Do Remote Administration and Browser-Based Log Review Fit Together? 181
How Can I Obtain a Printable Version of the EventReporter Manual? 182
Can EventReporter Write to a UNC Path? 182
Which Database Format Should I Use with EventReporter? 183
Can EventReporter Work with Custom Event Logs? 184
How Do I Enter EventReporter License Information? 184
Do the configuration clients require .NET Framework, or is .NET Core or .NET 5+ enough? 185
Is EventReporter v19+ supported on Windows Server IoT 2025? 186
Sales 186
How do I contact Adiscon sales? 187
What should I include in a quote request? 187
What happens after I open a sales ticket? 188
How do purchase orders and billing requests work? 189
Licensing and ordering 190
Air-gapped environments 191
Offline installation and activation 192
Online verification after activation 192
Perpetual licenses and UpgradeInsurance 193
UpgradeInsurance 194
Reference 194
EventReporter Shortcut Keys 195
Command Line Switches 195
Edition Comparison 195
Comparison of properties 196
Event Properties 196
Accessing Properties 197
System Properties 202
Custom Properties 203
Event-Specific Properties 204
Complex Filter Conditions 211
Connect to Computer 214
System Error Codes 215

Glossary 215
Database 216
Engine Only Install 217
IETF 218
IPv6 219
Registry File 220
Resource ID 221
RFC 3164 222
RFC 3195 223
RFC 5424 224
SETP 225
SMTP 226
Syslog 227
Syslog Facility 228
TCP 229
UDP 230
UTC 231
Copyrights 231

About EventReporter

EventReporter is a Windows event log collector and forwarding engine. It collects Windows Event Log data, filters it through rules, and then stores, forwards, or alerts on the events that matter.

EventReporter is designed for administrators who need more than the native Windows Event Viewer. It provides centralized forwarding, rule-based filtering, and background service operation for environments that want to integrate Windows Event Log data into syslog, SETP, email, files, databases, or other downstream systems.

Typical use cases include:

- forwarding Windows Event Log events to a central syslog server or SIEM
- sending selected events by email for operational alerting
- storing events in files or databases for retention and analysis
- filtering noisy event streams before they leave the Windows host
- running Windows event collection as a background service with a separate configuration client

EventReporter is focused on Windows Event Log collection. It complements, but is distinct from products such as WinSyslog, which focus on syslog reception.

Inside the product, collected data is configured through **services** that feed events into rulesets. In plain language, you can read those services as the configured inputs that collect Windows Event Log data and hand it to the rule engine.

For a neutral summary of how EventReporter and the other Adiscon Windows products can deliver data into ROSI-oriented deployments, see `shared/how-to-integrate-adiscon-windows-products-into-rosi`.

This manual is organized around first-time setup, task-oriented tutorials, configuration reference, FAQ content, and reference material.

Manual

Getting Started

Start here to understand what EventReporter is, how its main components work together, how Windows events move through the product, and how to build a first working configuration.

Installation

Use this page to install EventReporter and prepare the system for initial configuration.

Before you begin

Make sure that:

- the target system runs a supported version of Windows
- you have local administrative rights for the installation
- the system can install Microsoft .NET Framework 4.7.2 or a newer .NET Framework 4.x release for the Configuration Client if it is not yet present
- you know whether this system will run the full product or only the background service in a deployment scenario
- if this is a service-only target, the Configuration Client is not required on that system and the .NET Framework requirement applies where the client is installed and used

Supported platforms

Current EventReporter releases are intended for modern Windows versions, including Windows 10, Windows 11, Windows Server 2016, Windows Server 2019, Windows Server 2022, Windows Server 2025, and newer compatible releases.

For detailed platform notes, see System Requirements.

Installation steps

1. Download the current EventReporter installer from the EventReporter download page.
2. Run the installer with administrative rights.
3. Select the components you want to install.
    - The **EventReporter Service** is the runtime component.
    - The **EventReporter Configuration Client** is the administrative UI.
4. Finish the installer.
5. Start the EventReporter Configuration Client.
6. Create or review the initial configuration.
7. Apply the configuration so the EventReporter service uses the current settings.

What gets installed

A standard installation includes:

- the EventReporter service
- the EventReporter Configuration Client
- supporting program files and libraries
- local help content shipped with the product

What to do next

After installation, continue with:

- Understand the Components
- Collect Windows Events
- Creating an Initial Configuration

System Requirements

EventReporter is designed to run on current Windows client and server systems with modest resource requirements.

Supported operating systems

The current product is intended for:

- Windows 10
- Windows 11
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022
- Windows Server 2025
- newer compatible Windows releases

Do not assume that older end-of-life Windows versions are supported by current EventReporter releases. If you need documentation for legacy platforms, use the manual that matches the older product release.

Client requirements

The EventReporter Configuration Client:

- runs on supported 32-bit and 64-bit Windows environments where the current product is supported
- requires Microsoft .NET Framework 4.7.2 or a newer .NET Framework 4.x release, such as .NET Framework 4.8 or 4.8.1
- .NET Framework 4.x and .NET Core / .NET 5+ are different runtime families, so .NET Core and .NET 5+ do not satisfy this requirement
- the installer can add the required .NET Framework components if needed
- needs only modest disk space and memory for normal administration tasks

This requirement applies to systems where the EventReporter Configuration Client is installed. A service-only target does not require the Configuration Client.

Service requirements

The EventReporter service:

- runs as a Windows service in the background
- uses comparatively little memory and disk space in its base state
- consumes additional resources depending on queue size, action types, and event volume

Operational notes

- Database logging performance depends more on the database and DSN setup than on the EventReporter core service.
- Large queues, heavy filtering, or verbose debug logging will increase resource use.
- Additional Windows event channels available on server editions can also be monitored when the operating system exposes them.

Understand the Components

EventReporter is easiest to understand if you separate its main components by role.

The two main components

1. **EventReporter Service** This is the runtime component. It runs in the background as a Windows service, runs the configured input services, evaluates rules, and then stores or forwards matching events.
2. **EventReporter Configuration Client** This is the administrative user interface. You use it to define input services, rulesets, filters, and actions. Changes are made in the Configuration Client and then applied so the running EventReporter service can use the new configuration.

How they fit together

- The **EventReporter service** performs the actual collection, filtering, storage, and forwarding work through the configured input services.
- The **configuration client** defines what those input services, rulesets, and actions should do.
- The runtime service continues using the currently applied configuration until you save or apply changes from the client.

For the current split between remote administration and browser-based review, see How Do Remote Administration and Browser-Based Log Review Fit Together?.

What to read next

- To understand where EventReporter gets data, start with Collect Windows Events.
- To build a first working setup, continue with Creating an Initial Configuration.
- To understand rule processing, see Process and Filter.

Collect Windows Events

EventReporter collects Windows Event Log data through configured input services and turns it into internal events that can be filtered, stored, and forwarded.

What EventReporter receives

EventReporter is centered on Windows Event Log collection. In practice, this means:

- classic Windows Event Log sources
- modern Windows event channels exposed through the operating system

- custom event logs that Windows and the originating application register

Where to configure it

- Services is the entry point for event collection.
- Event Log Monitor V1 and Event Log Monitor V2 are the key EventReporter collection services.

Choosing a monitor version

Use the newer Event Log Monitor where it matches the target system and event channel requirements. Keep the older monitor only when you have a defined reason to use it for compatibility or legacy behavior.

Quick verification

- Configure one Event Log Monitor input service.
- Bind it to a ruleset with a visible action such as Write to File.
- Apply the configuration and confirm that new Windows events reach the target.

Creating an Initial Configuration

Use this page to build the first working EventReporter configuration: collect Windows Event Log events and write them to a local file.

In this manual, **input** is the plain-language concept, while the configured object is a **service**.

Goal

At the end of this procedure, EventReporter will:

- monitor one or more Windows event logs
- process matching events through a ruleset
- write them to a local file

Prerequisites

- EventReporter is installed.
- You can open the EventReporter Configuration Client.
- The EventReporter service is installed on the system.

Steps

1. Create a ruleset.
    - In the EventReporter Configuration Client, create a new ruleset.
    - Leave filtering simple for the first test so that visible events can match.
2. Add one file action to that ruleset.
    - Inside the ruleset, add a Write to File action.
    - Choose an easy-to-find test file path.
3. Create one event collection service.
    - Under Services, add an Event Log Monitor V2 service.
    - Bind that input service to the ruleset you created.
    - Select at least one Windows event log or channel to monitor.
4. Save and apply the configuration.
    - Apply or save the changes in the Configuration Client so the input service can use them.
    - Until you apply the changes, the running service continues to use the previous configuration.
5. Start or restart the EventReporter service if required.

How to verify

1. Trigger or wait for a Windows event that should be visible.
2. Confirm that the event is written to the configured file.
3. If nothing arrives, check:
    - the EventReporter service is running
    - the event collection service is enabled
    - the input service is bound to the correct ruleset
    - the file action is inside that ruleset
    - the selected event log or channel actually produces events

Expected result

If the configuration is correct, EventReporter reads Windows Event Log data and writes matching events to the configured file.

Next step

- To refine matching behavior, continue with Process and Filter.
- To forward events elsewhere, continue with Store and Forward.

Process and Filter

EventReporter uses a rules engine to decide what to do with each collected Windows event: keep it, drop it, store it, forward it, or trigger a follow-up action.

Where to configure

- Configuration explains the tree view and how services, rulesets, rules, filters, and actions fit together.
- Filter Conditions decide which events match a rule.
- Actions define what happens for matching events.

Recommended setup path

1. Start with one Event Log Monitor service bound to one ruleset.
2. Add one simple action, such as Write to File, so results are easy to verify.
3. Add filter conditions to narrow down the events you care about.
    - Start with event source, event ID, severity, or log name.
    - Add message-content filters only after the broad event path works.
4. Add further actions only after the rule matches exactly what you intend.

Things that commonly trip people up

- Rule order matters: rules are evaluated top-to-bottom inside a ruleset.
- The service-to-ruleset binding decides which ruleset sees a collected event.
- Defaults are templates. They do not process anything until you create an actual service or action instance.

Next steps

- Learn the underlying model in Core concepts.
- For the detailed tree structure, see Multiple RuleSets - Rules - Actions.

Store and Forward

Once EventReporter collects an event and it matches a rule, actions can store it locally or forward it to downstream systems.

Where to configure

- Actions is the entry point for output, storage, and notification behavior.

Common storage targets

- Flat files: Write to File
- Databases: Write to Database
- Windows Event Log: Write to Event Log

Database setup paths

- Default supported schema: Tutorial: Write Windows Events to Microsoft SQL Server
- Custom schema integration: Tutorial: Integrate EventReporter with a Custom Database Schema

Common forwarding targets

- Syslog: Forward Syslog
- SETP: Forward via SETP
- Email: Send Email

Related analysis component

- For reviewing stored logs later, Adiscon LogAnalyzer works with file- and database-based storage.
- Users who prefer a simpler standalone tool for direct log file review may also consider the third-party log viewer Retrospective by centeractive.

Recommended setup path

1. Start with one local storage action so results are easy to inspect.
2. Add forwarding after filtering behaves the way you expect.
3. Test remote targets early so host, port, authentication, and protocol mismatches are found before production rollout.

For browser-based review of stored data, use Adiscon LogAnalyzer as a separate component. For the current split between service administration and browser-based review, see How Do Remote Administration and Browser-Based Log Review Fit Together?.

Operate and Troubleshoot

After initial setup, most operational work is validating event intake, tuning rules, and diagnosing why something did or did not happen.

Quick checklist

1. Confirm the EventReporter service is running.
2. Confirm the Event Log Monitor service is enabled and bound to the intended ruleset.
3. Confirm your rule order and filters match what you expect.
4. Add a temporary Write to File action to inspect raw output quickly.

Useful diagnostics

- Export the current configuration and collect a debug log when investigating problems. See Tutorial: Export the Configuration and Create a Debug Log.
- If output to a remote system fails, verify connectivity, protocol settings, and credentials on both sides.

Where to look next

- Input issues: Services
- Matching issues: Filter Conditions
- Output issues: Actions
- Common operational questions: FAQ

Tutorials

Use this section for concrete EventReporter tasks. Each tutorial is self-contained and focuses on one outcome.

Tutorial: Prepare EventReporter Data for Adiscon LogAnalyzer

Use this tutorial when EventReporter should write data that you want to review later in Adiscon LogAnalyzer.

Goal

At the end of this procedure, EventReporter will write Windows events into a database that LogAnalyzer can open.

Recommended path

For EventReporter, the recommended LogAnalyzer path is database-backed storage. This avoids file-parser dependencies and is the most stable integration path in the current manual.

LogAnalyzer is the browser-based review component for stored data. It is not the EventReporter service administration interface. For that distinction, see How Do Remote Administration and Browser-Based Log Review Fit Together?.

Prerequisites

- A reachable database server
- An ODBC **System DSN** on the EventReporter host
- EventReporter access to the target database
- A LogAnalyzer deployment that is ready to connect to the same database

Steps

1. Complete Tutorial: Write Windows Events to Microsoft SQL Server so EventReporter writes matching events into the database.
2. Open **`../shared/tutorials/loganalyzer-setup-and-use`**.
3. Configure LogAnalyzer to use the same database as its data source.
4. Trigger one or more matching Windows events.

Verification

1. Confirm that EventReporter writes rows into the target table.
2. Open the configured source in LogAnalyzer.
3. Verify that the stored events appear there.

Next step

If you need to refine which events are stored before they appear in LogAnalyzer, continue with:

- Process and Filter
- Tutorial: Write Windows Events to Microsoft SQL Server

Complementary Tool: Retrospective

If you need a simpler way to search and analyze log files directly instead of using a web-based database setup, another available option is the third-party log viewer Retrospective by centeractive. It is suited for direct log file review without requiring a central database setup.

Tutorial: Collect Windows Events and Write Them to a File

Use this tutorial when EventReporter should collect Windows Event Log data and store matching events in a local text file.

Goal

At the end of this procedure, EventReporter will:

- monitor one or more Windows event logs
- pass matching events through a ruleset
- write them to a file on disk

Prerequisites

- A writable target directory for log files
- At least one ruleset for incoming events
- An Event Log Monitor service that will bind to that ruleset

Steps

1. Create or choose a ruleset.
2. Add a **Write to File** action.
    - Inside that ruleset, add a Write to File action.
3. Configure the target file.
    - Set **File Path Name** to the directory where EventReporter should write the files.
    - Set **File Base Name** to the logical file name prefix.
    - Keep the default extension unless you need something specific.
4. Create or choose an Event Log Monitor service.
    - Use Event Log Monitor V2 for new setups unless you have a compatibility reason to use V1.
    - Bind the service to the ruleset that contains the file action.
5. Save and apply the configuration.
6. Start or restart the EventReporter service if required.

Verification

1. Trigger or wait for a Windows event that should match.
2. Open the configured directory.
3. Confirm that EventReporter created or updated the expected log file.

Next step

If file logging works, continue with:

- Creating an Initial Configuration
- File Logging Options
- Store and Forward

Tutorial: Forward Windows Events to a Syslog Server

Use this tutorial when EventReporter should relay selected Windows events to a remote syslog receiver.

Goal

At the end of this procedure, EventReporter will forward matching Windows event log entries to a remote syslog server.

Prerequisites

- The destination host name or IP address
- The target port and transport mode expected by the receiver
- A ruleset that receives events from an Event Log Monitor service

Steps

1. Create or choose the ruleset whose events should be forwarded.
2. Add a Forward Syslog action to that ruleset.
3. Configure the remote target.
    - Enter the destination host name or IP address.
    - Set the destination port.
    - Choose the transport mode that matches the receiver.
4. Keep protocol choices conservative.
    - Use UDP only if message loss is acceptable.
    - Prefer TCP-based modes when reliable delivery matters.
5. Save and apply the configuration.
6. Restart the EventReporter service if required.

Verification

1. Trigger an event that matches the ruleset.
2. Confirm that the remote syslog server receives it.
3. If forwarding fails, verify host, port, protocol mode, and firewall rules on both sides.

Next step

If forwarding works, continue with:

- Syslog Forwarding
- Store and Forward

Tutorial: Forward Windows Events via TLS to rsyslog

Use this tutorial when EventReporter should forward selected Windows Event Log records to an rsyslog receiver over encrypted syslog transport.

Goal

At the end of this procedure, EventReporter will forward matching Windows events to an rsyslog server over TCP with TLS enabled.

Prerequisites

- The rsyslog receiver host name or IP address
- The TCP port used by the rsyslog TLS listener
- The framing mode expected by the receiver
- The CA certificate or client certificate files required by the receiver
- A ruleset that receives events from an Event Log Monitor service

Configure the rsyslog receiver

Configure the rsyslog server first so that it accepts TLS-protected syslog over TCP. The examples below use RainerScript syntax and follow the local rsyslog documentation under *../rsyslog2/doc/source/*.

Minimal TLS listener with anonymous authentication:

```
global(
  defaultNetstreamDriver="gtls"
  defaultNetstreamDriverCAFile="/etc/rsyslog.d/certs/ca.pem"
  defaultNetstreamDriverCertFile="/etc/rsyslog.d/certs/server-cert.pem"
  defaultNetstreamDriverKeyFile="/etc/rsyslog.d/certs/server-key.pem"
)

module(
  load="imtcp"
  streamDriver.name="gtls"
  streamDriver.mode="1"
  streamDriver.authMode="anon"
)

input(
  type="imtcp"
  port="6514"
)
```

Stricter listener with certificate validation and client name matching:

```
global(
  defaultNetstreamDriver="gtls"
  defaultNetstreamDriverCAFile="/etc/rsyslog.d/certs/ca.pem"
  defaultNetstreamDriverCertFile="/etc/rsyslog.d/certs/server-cert.pem"
  defaultNetstreamDriverKeyFile="/etc/rsyslog.d/certs/server-key.pem"
)

module(
  load="imtcp"
  streamDriver.name="gtls"
  streamDriver.mode="1"
  streamDriver.authMode="x509/name"
)

input(
  type="imtcp"
  port="6514"
  permittedPeer=["eventreporter01.example.net"]
)
```

Use *x509/name* when the receiver should validate the client certificate and restrict accepted senders to the permitted certificate names. Use *anon* only when that weaker trust model is acceptable in your environment. In both cases, restart rsyslog after changing the listener configuration.

Steps

1. Create or choose the ruleset whose events should be forwarded.

2. Add a Forward Syslog action to that ruleset.
3. Configure the target host and port.
    - Enter the rsyslog server host name or IP address.
    - Set the TCP port used by the rsyslog TLS listener.
4. Select a TCP-based syslog transport mode.
    - Prefer **TCP (octet-count based framing)** when the receiver supports it.
    - Otherwise use the TCP framing mode that matches the rsyslog listener.
5. Open the TLS settings for the action and enable **SSL / TLS Encryption**.
6. Select the TLS mode that matches the receiver configuration.
    - Use the default anonymous mode only if the receiver is configured for it.
    - Use certificate-based mode when the receiver expects CA validation or a client certificate.
7. If the receiver requires certificate-based trust, provide the matching files.
    - Select the common CA PEM file used to validate the receiver.
    - If mutual authentication is required, also select the client certificate PEM and key PEM files.
8. Keep protocol settings modern.
    - Leave **SSL v3** disabled.
    - Leave **TLS v1.0** disabled.
    - Use **TLS v1.2** or **TLS v1.3** when the receiver supports them.
9. Save and apply the configuration.
10. Restart the EventReporter service if required in your environment.

Verification

1. Trigger an event that matches the ruleset.
2. Confirm that the rsyslog receiver accepts the TLS connection and receives the forwarded event.
3. If forwarding fails, check:
    - target host and port
    - rsyslog *imtcp* listener configuration
    - TCP framing mode
    - CA, certificate, and key files
    - TLS version compatibility
    - *permittedPeer* entries on the rsyslog side when *x509/name* is used
    - firewall rules between EventReporter and rsyslog

Next step

If forwarding works, continue with:

- Syslog Forwarding
- Store and Forward

Tutorial: Forward Windows Events via SETP

Use this tutorial when EventReporter should forward selected Windows events to another Adiscon product over SETP.

Goal

At the end of this procedure, EventReporter will forward matching Windows event log entries via SETP.

Prerequisites

- A reachable SETP receiver
- The target host name or IP address
- Port and TLS settings agreed between sender and receiver

Steps

1. Create or choose the ruleset whose events should be forwarded.
2. Add a Forward via SETP action.
3. Configure the receiver connection.
    - Enter the destination host name or IP address.
    - Set the destination port.
    - Enable TLS only if the receiver is configured for it.
4. Save and apply the configuration.
5. Restart the EventReporter service if required.

Verification

1. Trigger a matching event.
2. Confirm that the SETP receiver gets the forwarded event.
3. If delivery fails, verify host, port, and TLS settings on both sides.

Next step

If the basic path works, continue with:

- Send SETP
- Store and Forward

Tutorial: Send Matching Windows Events by Email

Use this tutorial when EventReporter should send selected Windows events by email.

Goal

At the end of this procedure, EventReporter will send matching events as email messages through a configured SMTP server.

Prerequisites

- A reachable SMTP server
- A sender address accepted by that server
- At least one target recipient address

Steps

1. Create or choose the ruleset that should trigger email alerts.
2. Add a Send Email action.
3. Configure the SMTP connection.
    - Enter the mail server name or IP address.
    - Set the SMTP port.
    - Enable authentication if your server requires it.
    - Enable TLS or SSL only if the mail server supports it.
4. Configure sender and recipient fields.
5. Save and apply the configuration.
6. Restart the EventReporter service if required.

Verification

1. Trigger a matching event.
2. Confirm that the email reaches the recipient mailbox.
3. If it does not arrive, verify SMTP connectivity, authentication, and relay restrictions.

Next step

If the alert works, narrow the matching rule and review:

- Send Email
- Store and Forward

Tutorial: Write Windows Events to Microsoft SQL Server

Use this tutorial when EventReporter should store matching Windows events in Microsoft SQL Server through an ODBC **System DSN** and the built-in default database schema.

Goal

At the end of this procedure, EventReporter will write matching events into a default `SystemEvents` table in Microsoft SQL Server.

Why this tutorial uses the default schema

This is the fastest supported path for a first production deployment. It keeps the built-in field mapping, works with the **Create Database** button, and is the safest choice if you later want Adiscon-compatible tooling or predictable support behavior.

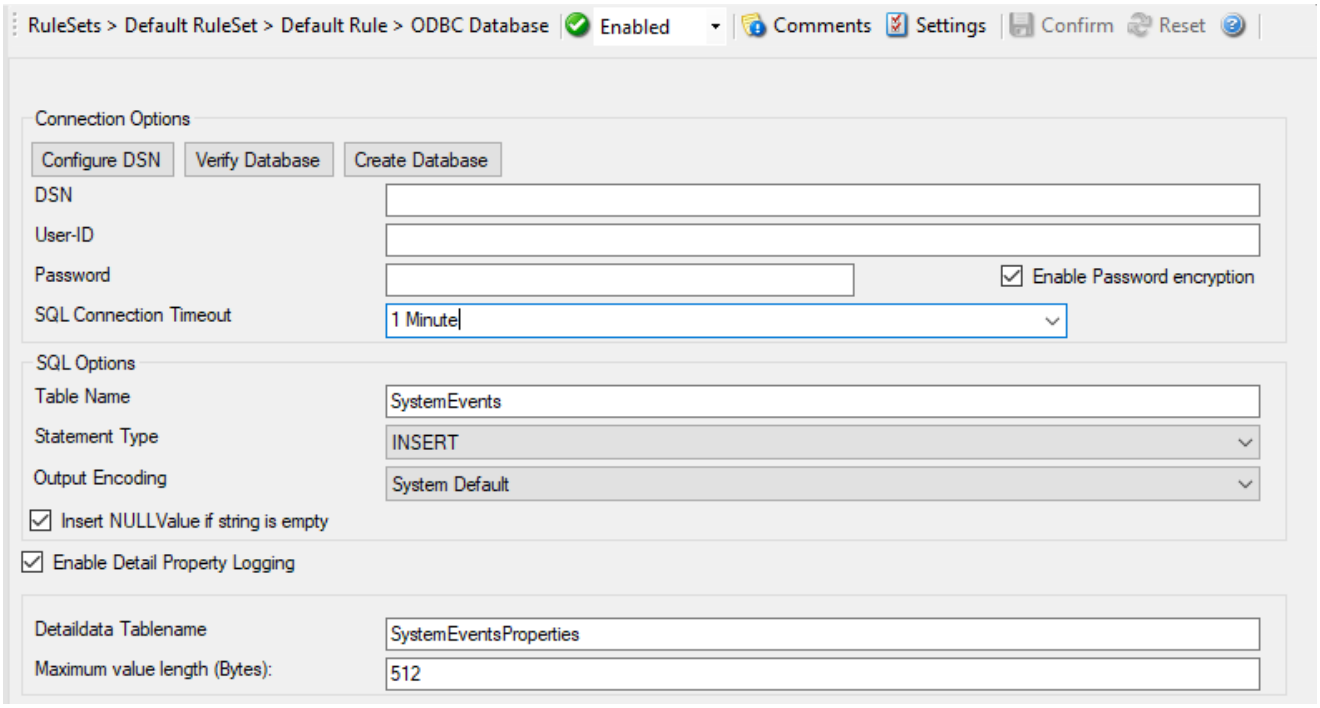
Prerequisites

- Microsoft SQL Server and SQL Server Management Studio (SSMS)
- Microsoft ODBC Driver 18 for SQL Server, or a compatible Microsoft SQL Server ODBC driver installed on the EventReporter host
- Database credentials with permission to connect, insert rows, and create the default tables
- Access to the EventReporter configuration client on the host where EventReporter runs

Steps

1. Create the target database in SQL Server.
    - Open SSMS and connect to the SQL Server instance that should receive the EventReporter data.
    - Create an empty database for this integration.
    - Keep the database name available for the ODBC setup.
2. Create and test an ODBC **System DSN** on the EventReporter host.
    - Open **Data Sources (ODBC)** on the EventReporter host.
    - Create a new **System DSN** for SQL Server.
    - Select the Microsoft SQL Server driver that matches your environment, for example Microsoft ODBC Driver 18 for SQL Server.
    - Point the DSN to the SQL Server instance and database created in the previous step.
    - Complete the DSN wizard and use its built-in connection test.
3. Create or choose the EventReporter ruleset whose events should be stored.
4. Add a Write to Database action to that ruleset.
5. Configure the database action.
    - Select the ODBC **System DSN** you created.
    - Enter the database credentials if the DSN or driver requires them.
    - Keep the default table name `SystemEvents`.

- Keep the default field list unless you intentionally need a custom schema.
- Leave the SQL statement type at the normal `INSERT` path unless you have a verified SQL Server stored-procedure design.



The connection tab is the main place to select the DSN, enter credentials, verify the connection, and create the default tables.

6. Create the default database tables.

   - Use the action's **Verify Database** button first.
   - If the connection test succeeds, click **Create Database**.
   - Confirm that EventReporter creates the default tables in the selected SQL Server database.

7. Save and apply the configuration.

   - Restart the EventReporter service if your environment or workflow requires it.

8. Trigger a matching Windows event.
9. Verify the inserted rows in SQL Server.

   - Open SSMS and query the `SystemEvents` table.
   - Confirm that the test event appears there.

```
SELECT TOP (10) *
FROM dbo.SystemEvents;
```

Verification

1. The ODBC **System DSN** test succeeds.
2. The action's **Verify Database** button succeeds.
3. The **Create Database** button creates the default tables.
4. A test event produces a new row in `SystemEvents`.

Common issues

- The DSN was created as a user DSN instead of a **System DSN**.
- The DSN points to the wrong SQL Server instance or database.
- The SQL Server account can connect but does not have permission to create tables or insert rows.
- The default field list or table name was changed even though the goal is the default supported schema.
- Another tutorial path is actually needed because the destination must be an existing custom table.

Next step

If the default schema path works and you want to keep it, continue with:

- ODBC Database Options
- Tutorial: Integrate EventReporter with a Custom Database Schema
- Store and Forward
- Which Database Format Should I Use with EventReporter?

Tutorial: Integrate EventReporter with a Custom Database Schema

Use this tutorial when EventReporter should write into an existing database schema instead of the built-in `SystemEvents` layout.

Goal

At the end of this procedure, EventReporter will write matching Windows events into your own destination table through an ODBC **System DSN**.

When to choose this tutorial

Use this path when:

- your organization already has a fixed database schema
- another application expects specific column names or data types
- you want EventReporter to feed an existing integration or reporting database

Do not use this path just because database logging exists. If you want the fastest supported setup or Adiscon-compatible default tables, use Tutorial: Write Windows Events to Microsoft SQL Server instead.

What this tutorial does not do

This tutorial does not design your schema for you. EventReporter can write to your table, but you must decide the table design, column definitions, indexes, retention strategy, and downstream reporting logic.

Prerequisites

- A reachable database server and a tested ODBC **System DSN**
- A destination table that already exists
- Database credentials with permission to insert rows into that table
- A clear mapping from EventReporter properties to the destination columns

Example target table

The exact schema is up to you. For a Microsoft SQL Server example, a custom table could look like this:

```
CREATE TABLE dbo.IncomingWindowsEvents (
    recorded_at datetime2 NOT NULL,
    source_host nvarchar(255) NOT NULL,
    event_source nvarchar(255) NULL,
    event_id int NULL,
    message_text nvarchar(max) NOT NULL
);
```

Steps

1. Review the destination schema before opening EventReporter.
    - Confirm the exact table name.
    - Confirm each destination column name and data type.

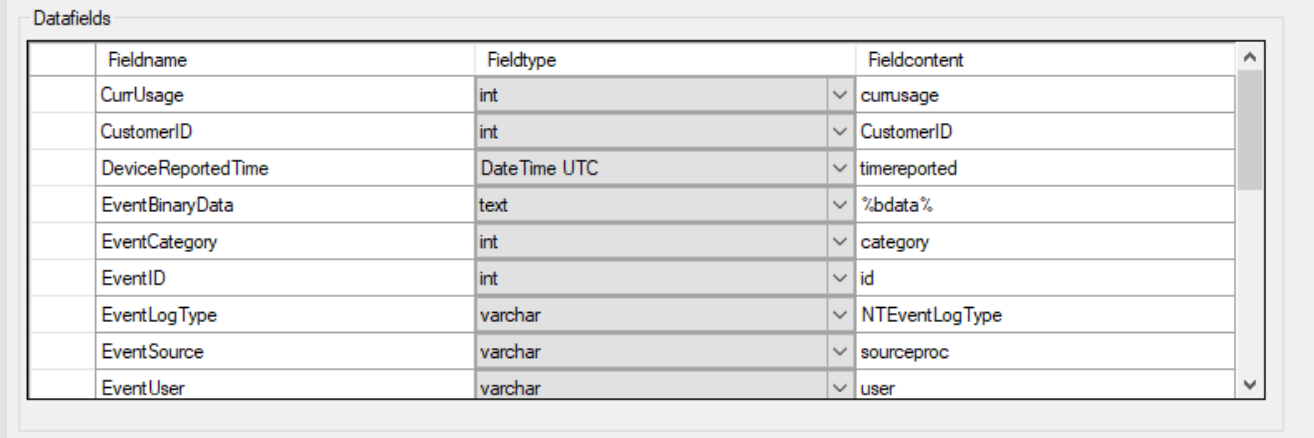
- Decide which EventReporter property belongs in each column.

2. Create and test an ODBC **System DSN** for the target database.
   - The DSN must point to the database that already contains your destination table.
   - Complete the DSN wizard and use its built-in test before you continue.
3. Create or choose the ruleset whose events should be stored.
4. Add a Write to Database action to that ruleset.
5. Configure the connection settings.
   - Select the ODBC **System DSN**.
   - Enter credentials if the DSN or driver requires them.
   - Use **Verify Database** to confirm connectivity.
6. Point the action to the existing destination table.
   - Enter the custom table name.
   - Do **not** use **Create Database** for this path unless you intentionally want the built-in default schema instead of your own table.
7. Replace the default field list with mappings that match your schema.

   For the example table above, a practical starting point is:

   - `recorded_at` -> `DateTime` -> `timegenerated`
   - `source_host` -> `varchar` -> `source`
   - `event_source` -> `varchar` -> `source`
   - `event_id` -> `int` -> `id`
   - `message_text` -> `text` -> `msg`

   If a string column is shorter than the source property, use the property replacer to truncate or transform the value deliberately. For example, `%msg:1:200%` stores only the first 200 characters of the message.



   The datafields tab is where you replace the default mapping with the column names, data types, and event properties that match your own schema.
8. Save and apply the configuration.
   - Restart the EventReporter service if your environment or workflow requires it.
9. Trigger a matching Windows event.
10. Query the destination table and verify the inserted data.

```
SELECT TOP (10)
    recorded_at,
    source_host,
    event_source,
    event_id,
    message_text
FROM dbo.IncomingWindowsEvents;
```

Verification

1. The ODBC **System DSN** test succeeds.
2. The action's **Verify Database** button succeeds.
3. A test event inserts a row into the existing custom table.
4. Each value appears in the expected destination column with the expected data type and length.

Common issues

- Leaving the default field list unchanged while targeting a custom table
- Using the wrong field type for a destination column
- Forgetting that long text may not fit into a short `varchar` column
- Clicking **Create Database** even though the goal is an existing custom schema
- Assuming Adiscon tools that expect the default schema will continue to work unchanged against the custom table

Next step

If the custom integration path works, continue with:

- ODBC Database Options
- Which Database Format Should I Use with EventReporter?

If you later decide that you need the built-in Adiscon table layout instead of your own schema, switch to Tutorial: Write Windows Events to Microsoft SQL Server.

Tutorial: Export the Configuration and Create a Debug Log

Use this tutorial when you need EventReporter configuration data and a debug log for troubleshooting or for a support case.

Goal

At the end of this procedure, you will have:

- a text-based export of the EventReporter configuration
- a debug log file that captures service behavior

Prerequisites

- Access to the EventReporter Configuration Client
- Permission to restart the EventReporter service if needed
- A writable location for the exported settings and debug log

Steps

1. Export the EventReporter configuration.
    - Open the EventReporter Configuration Client.
    - Go to **Computer -> Export Settings to Registry File**.
    - Export the settings as a text-based registry file.
2. Save the export to a known location.
    - Use a descriptive file name such as `eventreporter-config.reg`.
    - Do not use the binary export format.
3. Enable debug logging.
    - Open Debug under **General**.
    - Enable debug output into file.

   - Set a full path and file name for the debug log.
   - Start with **Errors & Warnings** and **Minimum Debug Output** unless support asks for a higher level.
4. Restart the EventReporter service if required.
5. Reproduce the problem or run the test scenario.
6. Disable debug logging again after capture.

   - Debug logging increases system load and should not stay enabled during normal operation.
7. Package the files for review or support.

   - Compress the exported registry file and the debug log file into one ZIP archive.

Verification

1. Confirm that the registry export file exists and is readable as text.
2. Confirm that the debug log file exists and contains recent entries from the test period.
3. If the debug log stays empty, verify that the service was restarted after enabling debug output.

Next step

If you need to send the data to Adiscon, continue with:

- How to Export EventReporter Settings for a Support Call
- How do I contact Adiscon sales?

Configuration

Use this section to configure how EventReporter collects Windows events, processes them through rules, and stores or forwards the results.

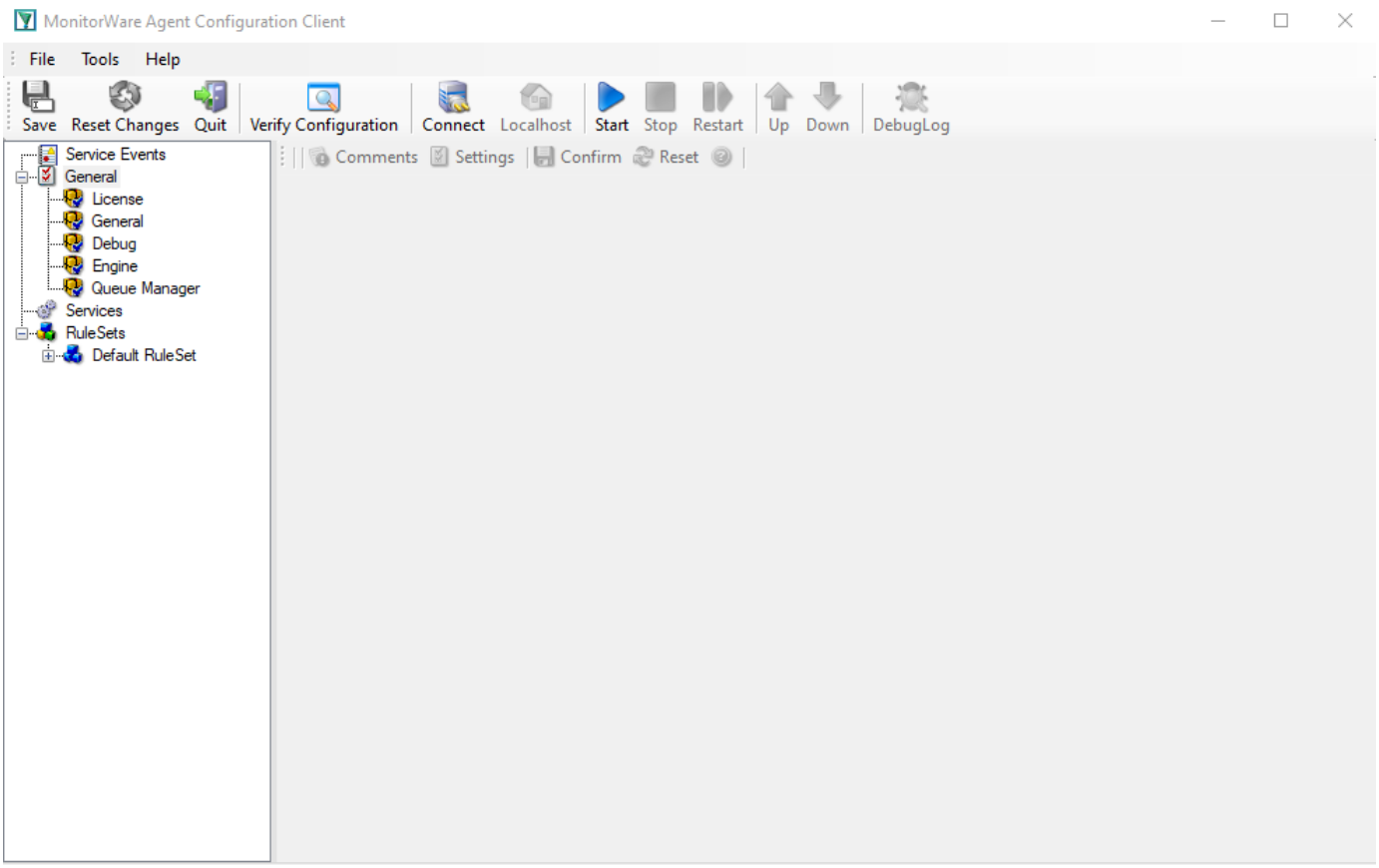
If you are new to the product, start with `Getting Started` for the first working setup and return here for the detailed configuration pages.

In this manual, **input** is the clearest plain-language concept for anything that collects events, while **service** remains the operational term for the configured EventReporter object.

Configuring EventReporter

This page explains how the EventReporter Configuration Client is organized and how its tree structure maps to the running EventReporter service.

The EventReporter service runs in the background after configuration. The Configuration Client is the administrative interface you use to define that configuration.



The tree structure

The Configuration Client is organized around three main areas:

- **General** for global options and defaults
- **Services** for the active event collection services
- **RuleSets** for filtering and actions

How configuration changes take effect

Configuration changes are made in the Configuration Client and then applied so that the EventReporter service can use them. Until you apply or save the changes, the running service continues using the previously active configuration.

Defaults versus instances

Defaults under **General** are templates. They do not collect, filter, or forward anything by themselves. They only provide default values for new service and action instances.

Services

Each configured service instance appears under **Services**. For EventReporter, this usually means one or more Event Log Monitor instances that collect Windows Event Log data and bind that data to a ruleset.

Rulesets, rules, filters, and actions

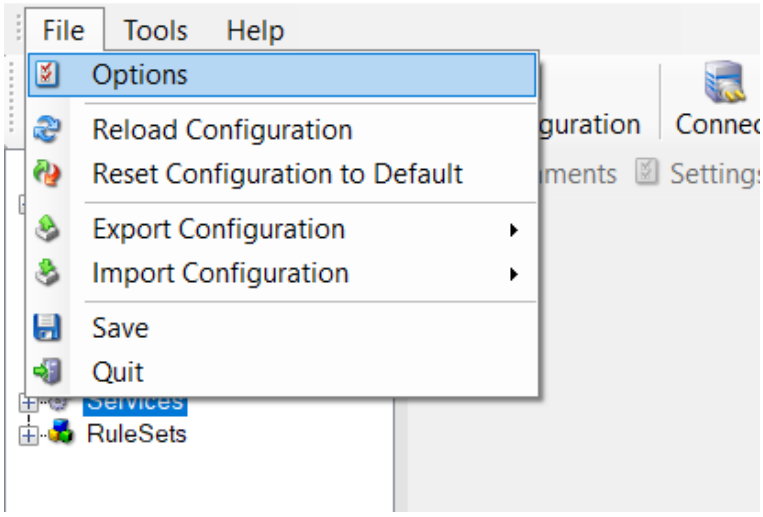
Each service sends its collected events to the ruleset configured for that service. Within the ruleset:

- rules are evaluated from top to bottom
- filter conditions decide whether a rule matches
- actions define what happens when the rule matches

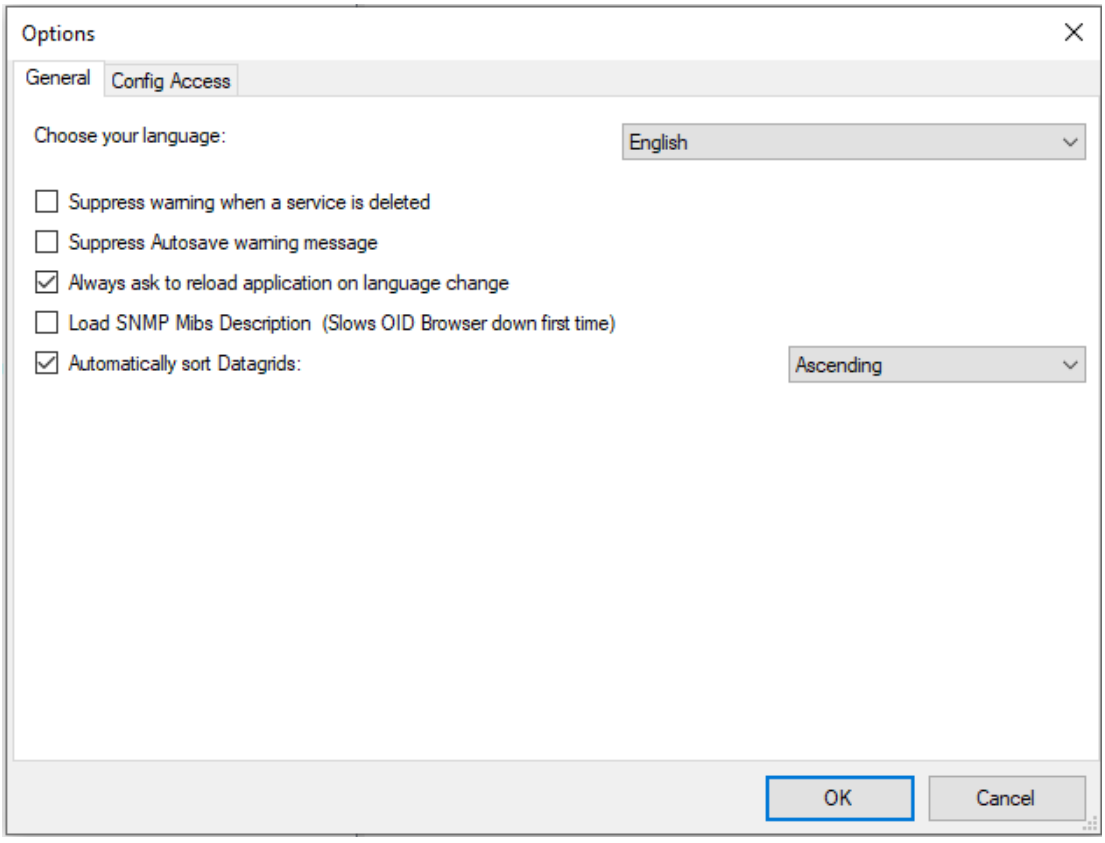
The following sections describe the detailed properties of each element.

Client Options

There are several options, that refer to the configuration client and not to the service. These can be found under File -> Options



- Client Options*



- General Tab*

Choose your language

You can choose a language pack. “English” is the default and suggested language.

Suppress warning when a service is deleted

If this option is checked you will not get a warning when you try to delete a service and there is no other service that uses the connected ruleset.

Suppress autosave warning message

If you make changes in the configuration and switch to another component, a warning will occur if you haven’t saved the changes. This warning will also allow you to directly enable auto-saving the configuration.

Always ask to reload application after language change

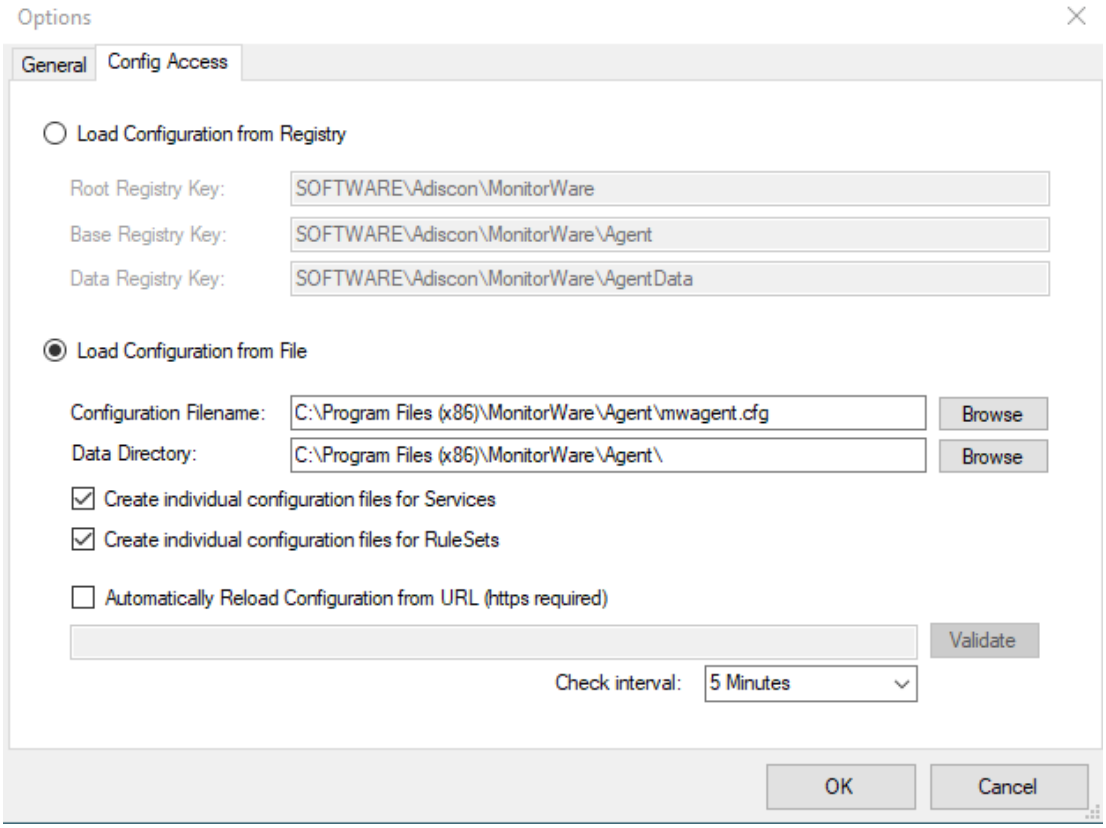
When you change the language, a popup will ask you to reload the configuration client to properly apply the changes and load with the set language.

Load SNMP Mibs Description (Slows OID Browser down first time)

If enabled, load SNMP Descriptions from MIB files (Client starts a little slower on startup).

Automatically sort Datagrids

Datagrids are used in certain areas within the configuration objects. You can change the default sorting behavior from ascending to descending here.



- Config Access Tab*

Load Configuration from Registry

The Configuration Client can be switched to a different registry path for configuration. The registry path change can be made permanent here. The changed registry path is the saved within the Parameters key of the Service.

Load Configuration from File

Alternatively, you can configure the service to load the configuration from a file. You can set the paths with the two fields below.

When enabled, the configuration will always be backed up before applying the new configuration. The backup consists of the last iteration and will be placed in the same directory.

Create individual configuration files for Services

Can only be enabled when “Load Configuration from File” is enabled. When enabled, the Services section of the configuration will be put into a separate file.

Create individual configuration files for RuleSets

Can only be enabled when “Load Configuration from File” is enabled. When enabled, the RuleSet section of the configuration will be put into a separate file.

Automatically Reload Configuration from URL (https required)

Only possible if File Configuration Mode is used.

If enabled, the configuration will be reloaded from a remote https location. Please note that a valid SSL certificate is required, or if custom certificates are used they have to be imported on the local machine properly.

If the remote configuration file can be downloaded from the configured location and differs from the current configuration, it will be installed automatically and the service will reload itself.

Check interval

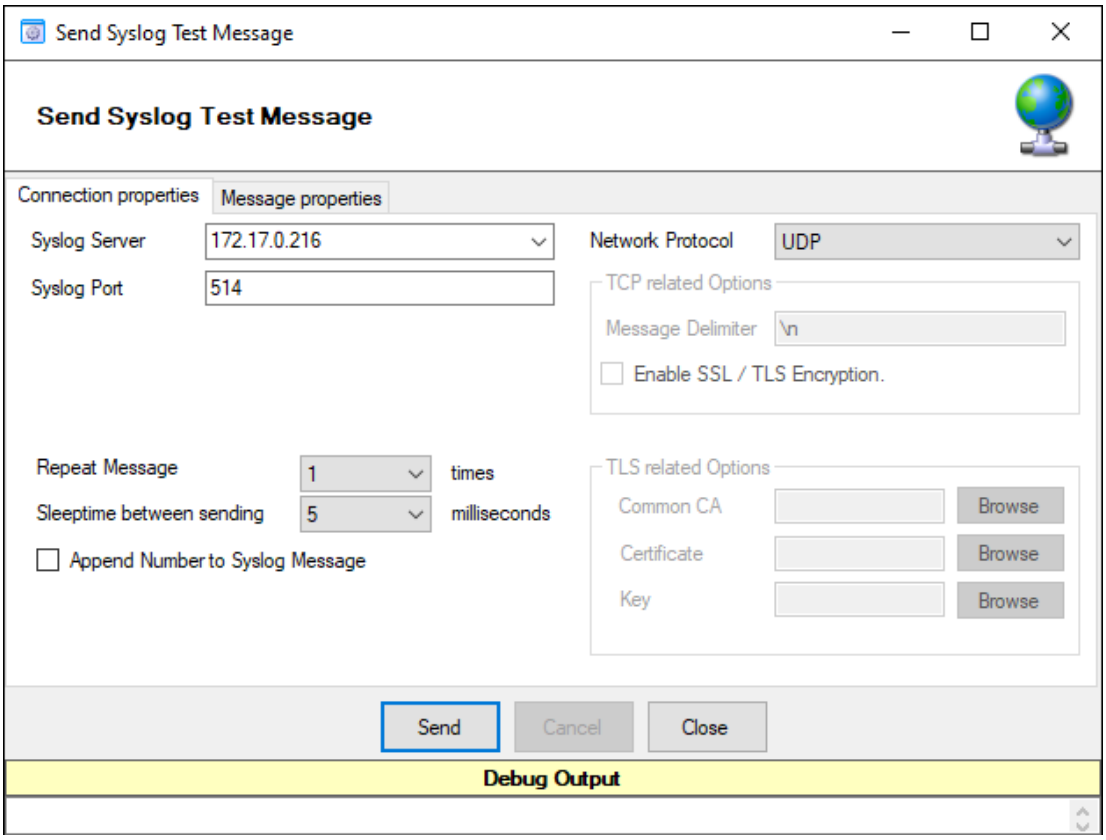
Specifies how often the service will check for remote configuration files. Please keep in mind that the configuration needs to be downloaded each time from the remote https url for comparison with the local one. We do not recommend to use a value lower then 5 minutes.

Client Tools

There are tools within the configuration client that you can use to test certain services or debug the application in general. Some can be found in the Tools menu.

Syslog Test Message

Opens a new windows which can send syslog test messages to Syslog Servers. This can also be opened within the configuration window of a Syslog service.



- Syslog Test Message Connection properties - UDP*

Syslog server

The hostname or ip address of the target Syslog server.

Syslog Port

The port that should be used to connect to the target Syslog server.

Repeat Message

How often you want to repeat the test message. Can be configured from 1 to 1000.

Sleeptime between sending

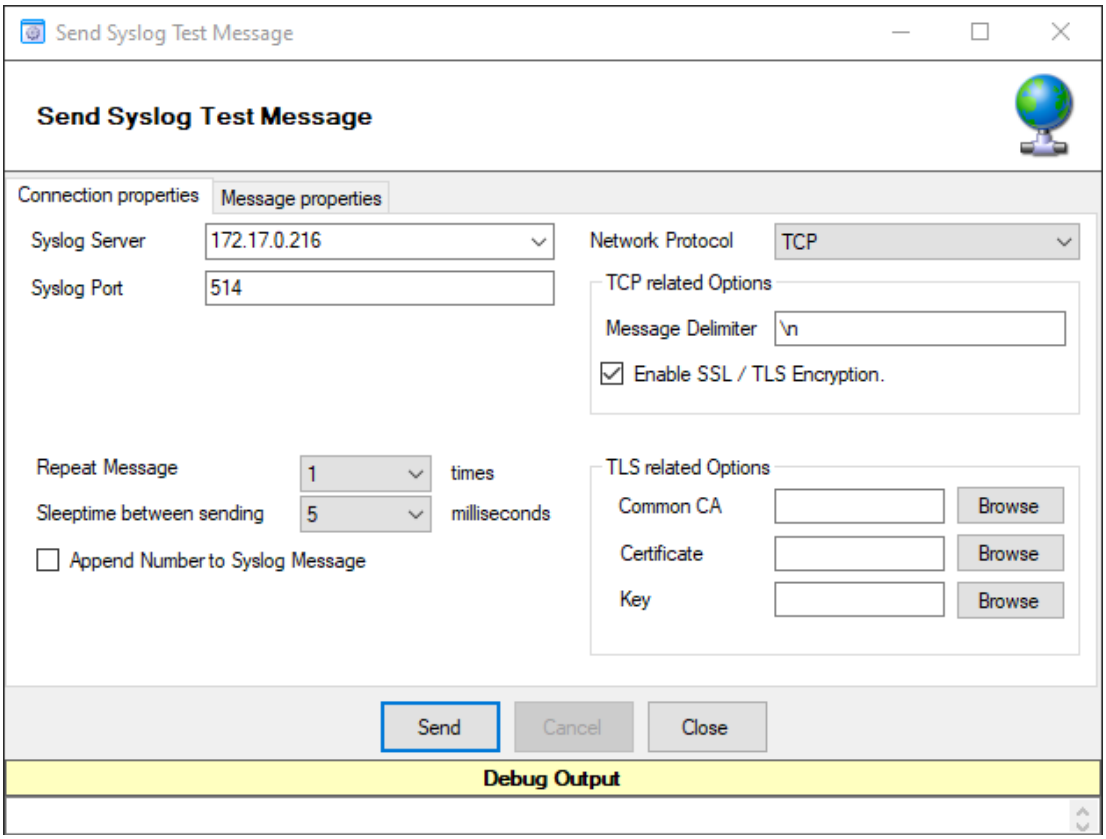
When using TCP, you can use 0ms. For UDP we recommend 1-5ms as sleeptime between sending syslog messages. Otherwise package loss can happen.

Append Number to Syslog Message

If sending multiple messages, enable this option in order to add a syslog number at the end of the message.

Network Protocol

Which network protocol should be used, either UDP or TCP can be selected.



- Syslog Test Message Connection properties - TCP*

Message Delimiter (TCP related Options)

When using TCP protocol, a message delimiter (separator) can be configured which is a simple linefeed by default.

Enable SSL/TLS Encryption (TCP related Options)

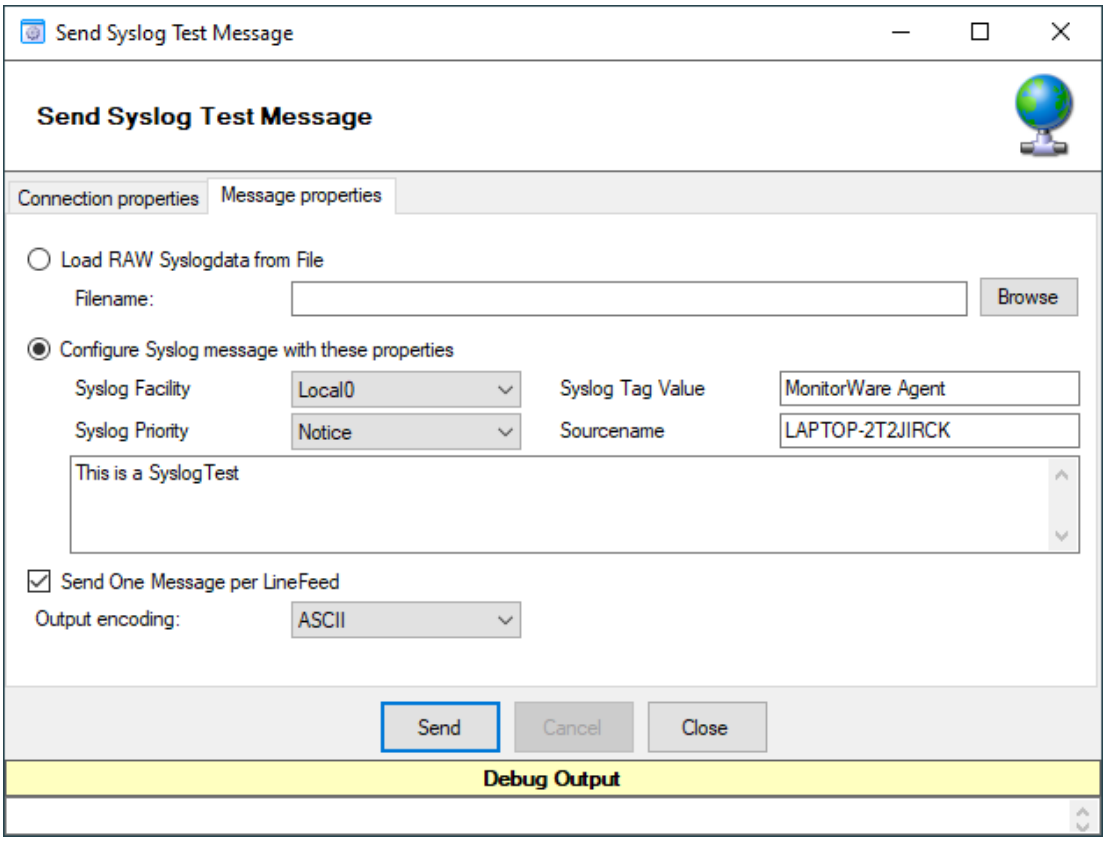
Check this option to enable the TLS related Options.

TLS related Options (TCP related Options)

Select common CA: Select the certificate from the common Certificate Authority (CA), the syslog receiver should use the same CA.

Select Certificate: Select the client certificate (PEM Format).

Select Key: Select the keyfile for the client certificate (PEM Format).



- Syslog Test Message Message properties*

Load RAW Syslogdata from File

You can choose to load raw syslogdata from file using this option. When loading UTF8 data make sure to set the Output encoding format from ASCII to UTF8. And if your file contains multiple syslog messages make sure that - Send One Message per LineFeed - is checked.

Configure Syslog message with these properties

Choose this if you want to configure all properties of the syslog message manually.

Send one Message per LineFeed

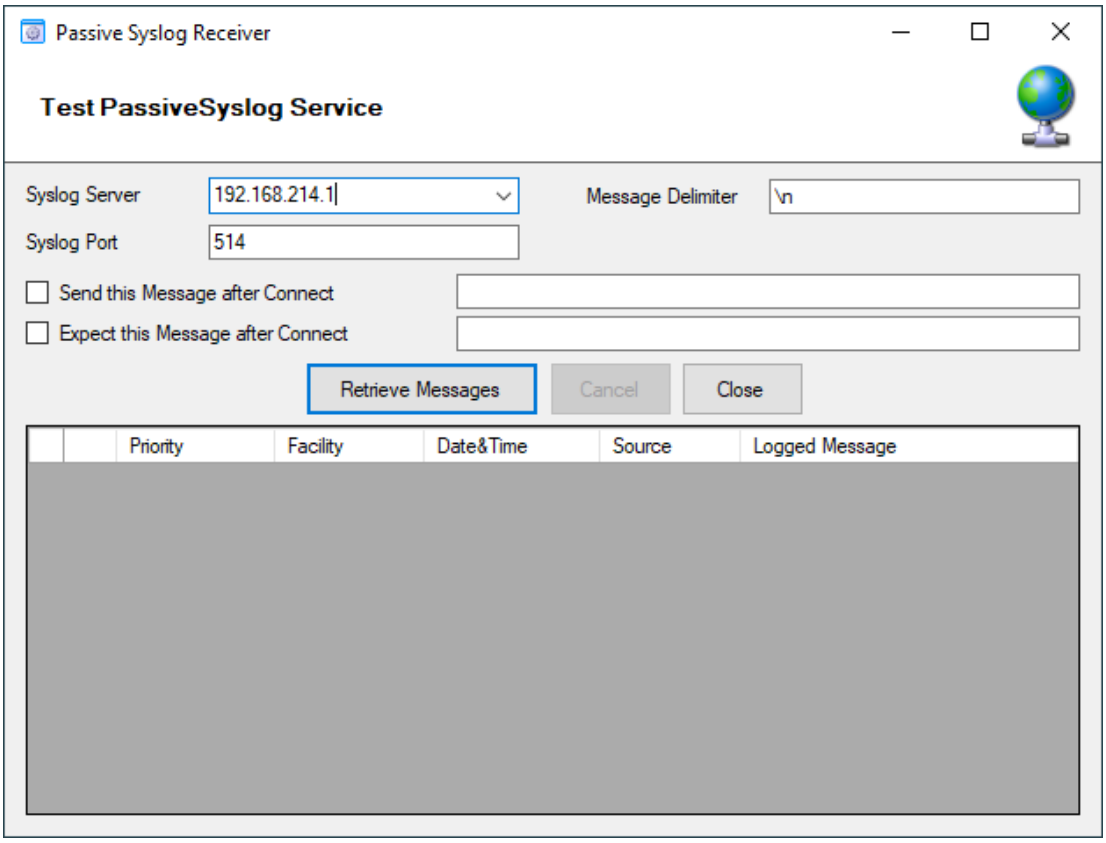
Check if your syslogdata contains multiple syslog messages divided by line feeds

Output encoding

Select the Output encoding you wish to use. When using UTF8, the UTF8 BOM is automatically prepended.

Passive Syslog Receiver

Opens a new windows to test Passive Syslog Servers. This can also be opened within the configuration window of a Passive Syslog service.



- Test Passive Syslog Service*

Syslog server

The hostname or ip address of the target passive Syslog server.

Syslog Port

The port that should be used to connect to the target passive Syslog server.

Message Delimiter

The message delimiter (separator) used to split syslog messages which is a simple linefeed by default.

Send this Message after Connect

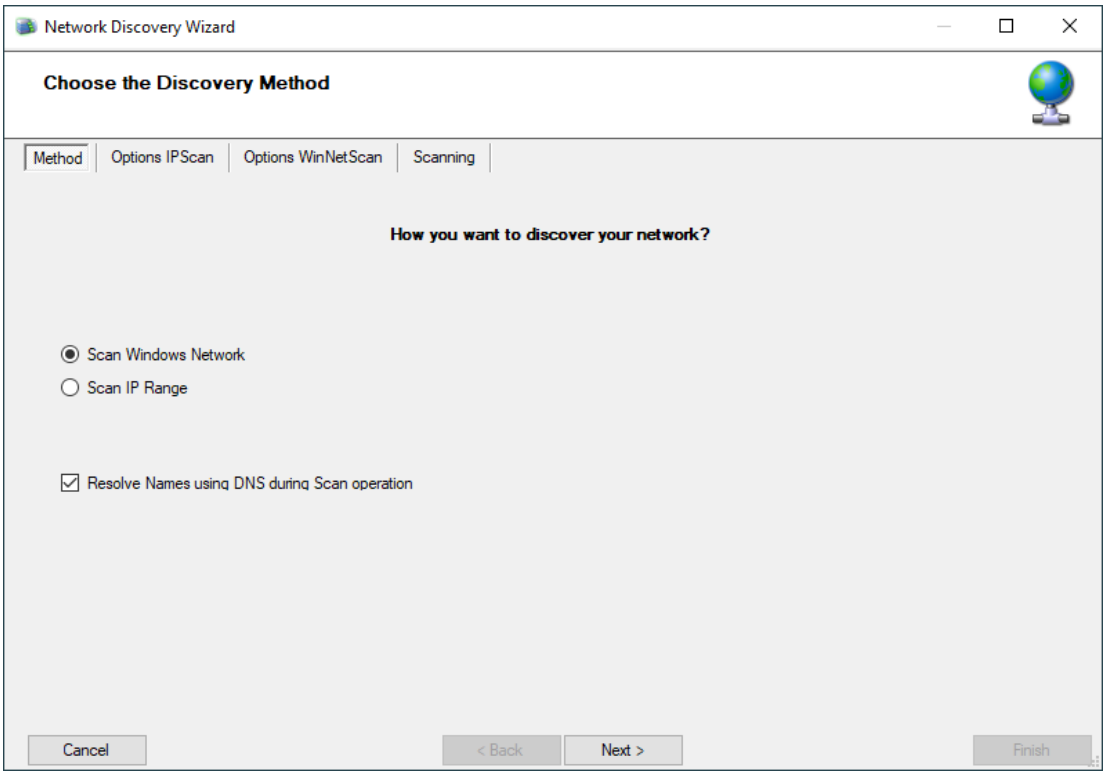
If required, configure a custom message that is send to the server after connect.

Expect this Message after Connect

If required, configure a custom message that is expected by the sender when the server response to our custom message.

Network Discovery

Opens up a Wizard that will help you discover devices in your local network. Once the wizard has scanned your network, it will show Windows compatible devices it has found. Please note that this will require Windows Management Instrumentation (WMI) access to the remote machines which may be disabled in Windows Firewalls by default.



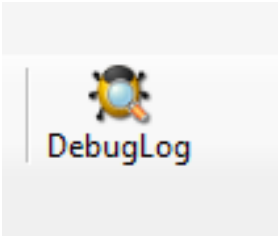
- Network Discovery - Choose the discovery Method*

Kill Service

When stopping a service, and it does not shutdown in the time period, you can use this function to forcefully stop the service. The service process will be killed if possible.

DebugLog

The **DebugLog** Button will be available if Debug Logging is enabled in your Debug Options



- DebugLog*

When clicked, a new Logviewer window will be opened. The Debug Logviewer can load, parse, and analyze debug log-files from the service.

- Logviewer (Debug)*

Debugfile

Will automatically be set to your configured debug file. You can also choose other saved debug-files for analysis.

Load

When Load is clicked, the Logviewer will load lines as configured in the initial debug-lines field. When loading all log-lines on a large debug log-file, this may take a while. While the Load button is grayed out, the Logviewer will continue to read data from the debug log as it is being written.

Stop

Stop continuous loading of the debug log.

Reset

Will reset all loaded log-lines from memory and clear the debug data-grid.

Init Debuglines

The amount of log-lines you want to read the first time.

Show debug messages for

Once the debug-log is processed, the Logviewer will automatically add filters for objects like services, rulesets, rules, and actions. You can use this select box to filter by them.

Filtering (bottom bar)

At the bottom of the Logviewer window, you can filter the debug-log for Thread (ID), Priority, internal Facility, and Functions. You can also filter for words or word sequences. The view will automatically be refreshed once you changed a filter.

Using File based configuration

Working with File based Configurations

Support for running the Service from file based configuration may be interesting for environments where you want to minimize registry access to a minimum or you want to manually edit the configuration without using the configuration client every time.

The Adiscon Configuration format is quiet simple. In the following description, all the configuration options will be explained in detail.

Adiscon Configuration format explained

Our configuration format is something between JSON and XML but hold at a very simple level.

Variables

All variables start with a dollar ($). Name and Value of a variable are separated by the FIRST space character. Everything else behind the first space will be considered as the Value. A line feed terminates the value. If your configuration value contains line feeds, you have to replace them with “\n” or “\r\n”. A single backslash can be used to escape brackets ({ and }).

Comments

All lines starting with a sharp (#) at the beginning will be ignored.

File Includes

Sample

```
includeconfig my-subconfigfiles-*.cfg*
```

The includeconfig statement will include either a single file or many files based on a filename pattern. In this sample all Files starting with “my-subconfigfiles-” and ending with “.cfg” will be included into the configuration. It is possible to create your own custom file structure with includes. The configuration client will be able to load and show your custom file structure, however it will not be able to maintain (save) it. We support a maximum include depth of up to 10 levels when using the includeconfig statement.

General Options

Sample

```
general(name="[name]") {
 $nOption 1
 ...
}
```

All options between the brackets will be loaded as variables into the general configuration object. The name attribute field specifies the general configuration block name. The brackets start and end an object block.

Services

All possible configuration parameters are named within the detailed services documentation.

Sample Service configuration:

```
input(type="[ID]" name="[name]") {
 $var1 Value1
 $var2 Value2
 ...
}
```

The brackets start and end a service block. All variables between the brackets will be loaded into the service configuration. The name attribute specifies the service display name. The type attribute contains the service type ID. It can be one of the following types:

```
1       = Syslog
2       = Heartbeat
3       = EventLog Monitor V1 (Win 2000 / XP / 2003 )
4       = SNMP Trap Listener
5       = File Monitor
8       = Ping Probe
9       = Port Probe
10      = NTService Monitor
11      = Diskspace Monitor
12      = Database Monitor
```

```
13      = Serialport Monitor
14      = CPU Monitor
16      = MonitorWare Echo Request
17      = SMTP Probe
18      = FTP Probe
19      = POP3 Probe
20      = IMAP Probe
21      = IMAP Probe
22      = NNTP Probe
23      = EventLog Monitor V2 (Win VISTA/7/2008 or higher)
24      = SMTP Listener
25      = SNMP Monitor
26      = RELP Listener
27      = Passive Syslog Listener
1999998 = MonitorWare Echo Reply
1999999 = SETP Listener
```

RuleSets

All possible configuration parameters are named within the detailed actions documentation.

Sample

```
ruleset(name="[name]" expanded="[on/off]") {
 rule(name="[name]" expanded="[on/off]" actionexpanded="[on/off]"
 ThreatNotFoundFilters="[on/off]" GlobalCondProperty="[on/off]"
 GlobalCondPropertyString="" ProcessRuleMode="[0/1/2]"
ProcessRuleDate="[uxtimestamp") {
       action(type="[ID]" name="[name]") {
               $var1 Value1
               $var2 Value2
               ...
       }
       filter(nTabSelection="0") {
               $nOperationType AND
               $PropertyType NOTNEEDED
               $PropertyValueType NOTNEEDED
               $CompareOperation EQUAL
               $nOptionalValue 0
               $nSaveIntoProperty 0
               $szSaveIntoPropertyName FilterMatch
       }
    }
}
```

The brackets start and end a ruleset block. The attributes of a Ruleset are self-explainable. Within a RuleSet, you can have Rules. The attributes of Rules are also self-explainable and partially Global Conditions that are equal to the options found in the Filter dialog. Within a Rule you can one Basefilter. This Basefilter again can have child filters it and these child filters can have child filters again. All “expanded” settings are optional and only important for the client treeview.

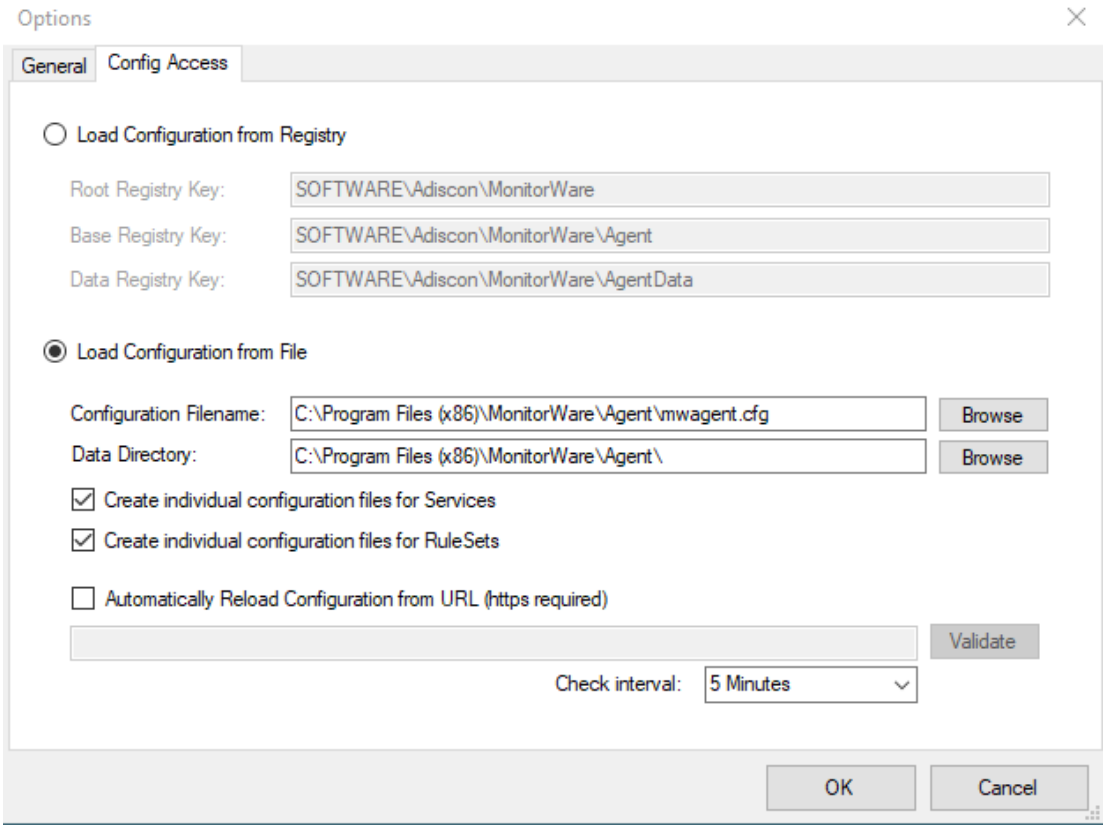
Within a Rule you can have Actions. The brackets start and end an action block. All variables in an action block between the brackets will be loaded into the action configuration. The name attribute specifies the service display name. The type attribute contains the action type ID. It can be one of the following types:

```
1000 = ODBC Database
1001 = Send Syslog
1008 = Net Send
1009 = Start Program
1011 = Send SETP
1012 = Set Property
1013 = Set Status
1014 = Call RuleSet
1015 = Post Process
1016 = Play Sound
1017 = Send to Communication Port
```

```
1021 = Send SNMP
1022 = Control NT Service
1023 = Compute Status Variable
1024 = HTTP Request
1025 = OleDB Database
1026 = Resolve Hostname
1027 = Send RELP
1028 = Send MS Queue
1029 = Normalize Event
1030 = Syslog Queue
```

How to enable file based configuration?

To switch from registry to file configuration mode, all you need to do is to go the “Config Access” tab in the Configuration “Client Options” and switch from “Load Configuration from Registry” to “Load Configuration from File” mode. Once you accept the change, the Client will ask you if you want to export the current loaded configuration into the file. Hit YES if you want to do so and NO if already have an existing configuration file. The configuration client will reload itself automatically after this.



- Client Options Configure File Based Configuration*

Create individual configuration files for Services

When enabled, the configuration client will create separated configuration files for each configured service. The main configuration file will then use the includeconfig statement to include all these configuration files by using a pattern. When deleting a service, its configuration file will be deleted as well.

Create individual configuration files for RuleSets

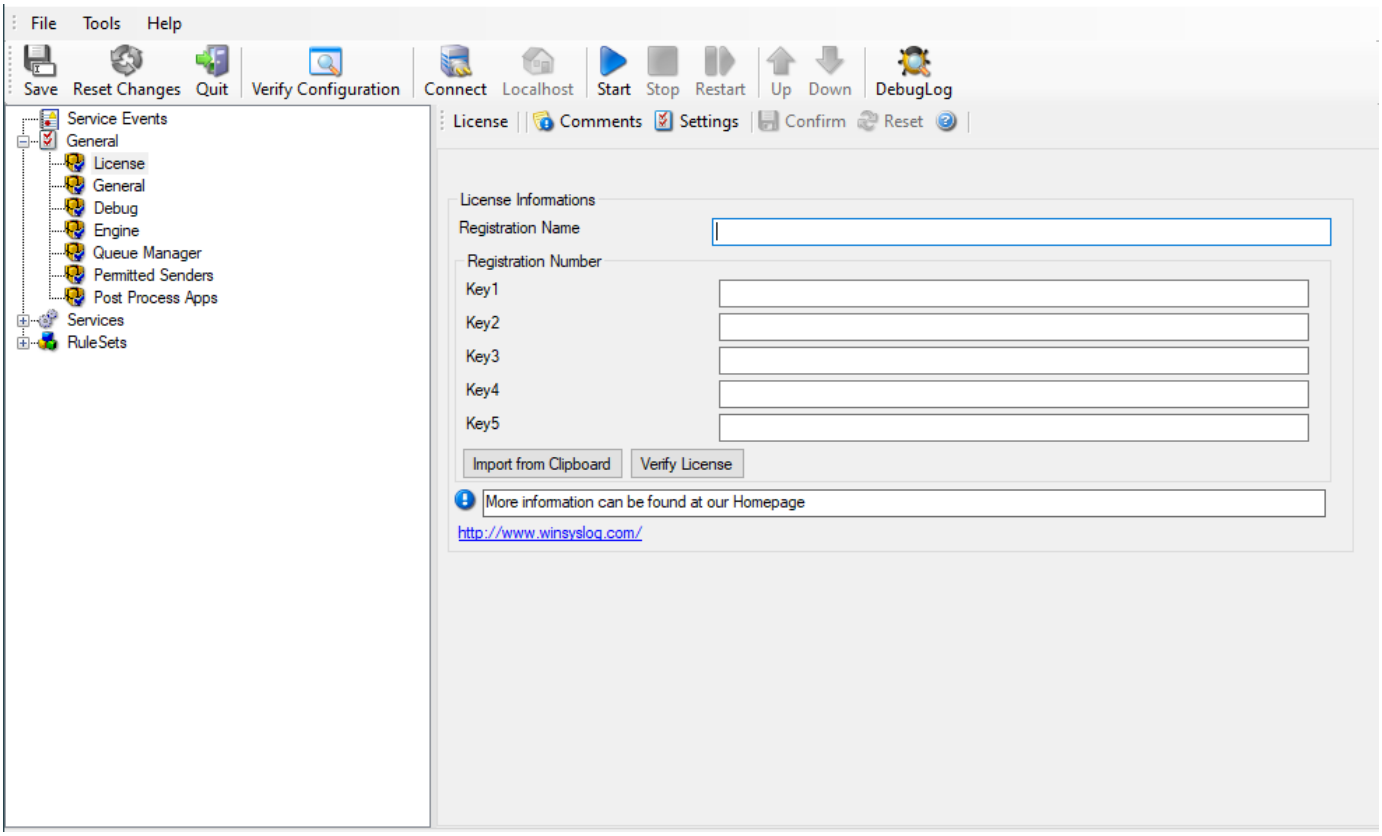
When enabled, the configuration client will create separated configuration files for each configured ruleset. The main configuration file will then use the includeconfig statement to include all these configuration files by using a pattern. When deleting a ruleset, its configuration file will be deleted as well

General Options

In this chapter, you find the general option settings.

License

After the purchase, the licensing information can be entered here.



Registration Name

File Configuration field:

szlicense

Description

The user chooses the registration name. It should correspond to your organization name, e.g. a company called “AA Carpenters, Inc.” should not choose “AA” as registration name. This can easily be mistaken and most probably be rejected by Adiscon for that reason. With the above scenario, we recommend using the full company name “AA Carpenters, Inc.”.

Please note: The registration name is case sensitive. It must be entered exactly as given. Leading and trailing spaces are also part of the registration name, so be sure to enter none.

Registration number

File Configuration field:

nLicenseKey1, nLicenseKey2, nLicenseKey3, nLicenseKey4, nLicenseKey5

Description

Adiscon provides this number. It is valid for a specific registration name. Be sure to enter the correct registration number. Each block of the license key must be filled into one of the key fields. Alternatively, you can use the “Import from Clipboard” button. The client detects invalid registration numbers and reports the corresponding error.

Import from Clipboard

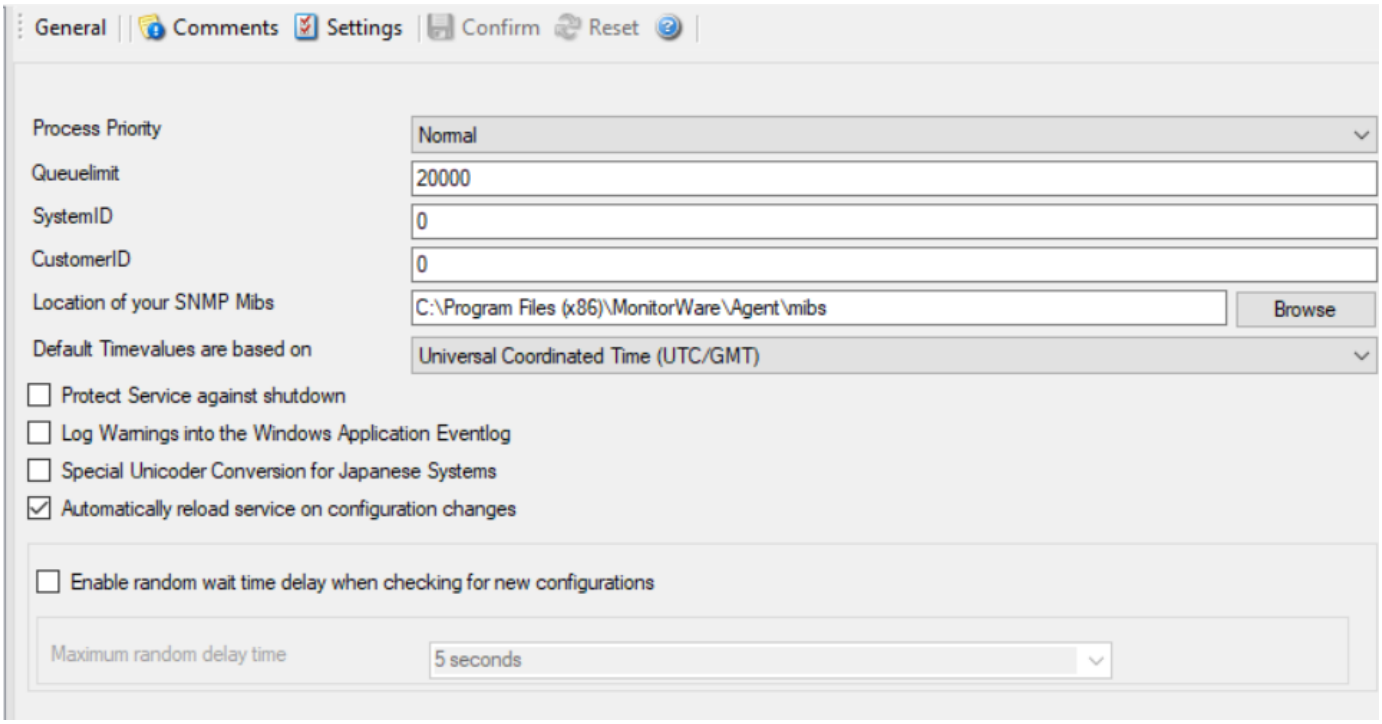
If the key has been copied to the clipboard it can be imported with this button.

Verify License

Here it can be verified if the license is valid.

General

The General Options available on this form are explained below:



Process Priority

File Configuration field:

nProcessPriority

Description

Configurable Process Priority to fine-tune application behavior.

Queuelimit

File Configuration field:

nQueueLimit

Description

The applications keeps an in-memory buffer where events received but not yet processed are stored. This allows the product to handle large message bursts. During such burst, the event is received and placed in the in-memory queue. The processing of the queue (via rulesets) itself is de-coupled from the process of receiving. During traffic bursts, the queue size increases, causing additional memory to be allocated. At the end of the burst, the queue size decreases and the memory is freed again.

Using the queue limit, you can limit that maximum number of events that can be in the queue at any given time. Once the limit is reached, no further enqueuing is possible. In this case, an old event must first be processed. In such situations, incoming events might be lost (depending on the rate they come in). A high value for the queue size limit (e.g. 200,000) is recommended, because of the risk of message loss.

It is also possible to place no limit on the queue. Use the value zero (0) for this case. In this case, the queue size is only limited by virtual memory available. However, we do not recommend this configuration as it might cause the product to use up all available system memory, which in turn could lead to a system failure.

SystemID

File Configuration field:

nSystemID

Description

SystemID is of type integer to be used by our customer. In addition, it is user configurable.

CustomerID

File Configuration field:

nCustomerID

Description

CustomerID is of type integer provided for customer ease. For example if someone monitors his customer's server, he can put in different CustomerIDs into each of the clients. Let us say someone monitors servers A and B. A has 5 servers all of them with CustomerID = 1 and B has 2 servers all of them with CustomerID = 2. Both A and B happen to have a server named "SERVER". Together with the customerID, these machines are now uniquely identifiable. This is user configurable.

Location of your SNMP MIBs

File Configuration field:

szMIBSPath

Description

Click the Browse button to search for your MIBs location or enter the path manually. The Client and Service will read all files from this directory automatically on startup.

Default Timevalues are based on

File Configuration field:

nTimeMode

Description

The general options of each product (EventReporter, MonitorWare Agent and WinSyslog) contain a setting for the "Default Timevalues are based on". This setting can be set to Localtime and UTC (Universal Coordinated Time) which is default. This setting has an effect on:

- Send Email Action: The date in the email header is affected
- Start Program Action: Time parameters in the command line are affected
- Write File Action: Time properties in the file name are affected
- Filter Engine: If you filter by weekday or time fields, localtime does affect the filter result

For information about local time output, see **`FAQ: default time values explained`**.

Protect Service against shutdown

File Configuration field:

nProtectAgainstShutdown

Description

When enabled, the Agent will not stop processing the internal queue when it is stopped. **Please note that it will remain in the stopping state then.**

Log Warnings into the Windows Application Eventlog

File Configuration field:

nEnableEventlogWarnings

Description

The Service will also log Warnings into the Windows Application Eventlog, and so be more verbose for troubleshooting. Default is disabled.

Special Unicoder Conversion for Japanese Systems

File Configuration field:

nJapanStringHandling

Description

This is a historical option for older multibyte systems from the time when UTF8 was not known yet. If enabled, whenever text is being converted from 16 Bit wide character to 8 Bit character, the conversion is done with bit masking in order to avoid broken encoding. **For today modern systems, we do NOT recommend to enable this option.**

Automatically reload service on configuration changes

File Configuration field:

nEnableAutoConfigReload

Description

When enabled (default), the service will detect configuration changes and reload its core automatically. This feature only works if the latest Client Application is used for configuration. It will also work if you are using the file based configuration method and update the configuration file. It will not work if you are using the service in console mode unless you send any input to the console.

Enable random wait time delay when checking for new configurations

File Configuration field:

bAutoReloadRandomDelay

Description

When enabled, a random delay (with the configured maximum) will be added between new configuration checks.

Maximum random delay time

File Configuration field:

nAutoReloadDelayTime

Description

The maximum for this random delay is 24 hours. The random delay has no affect on the service control anymore.

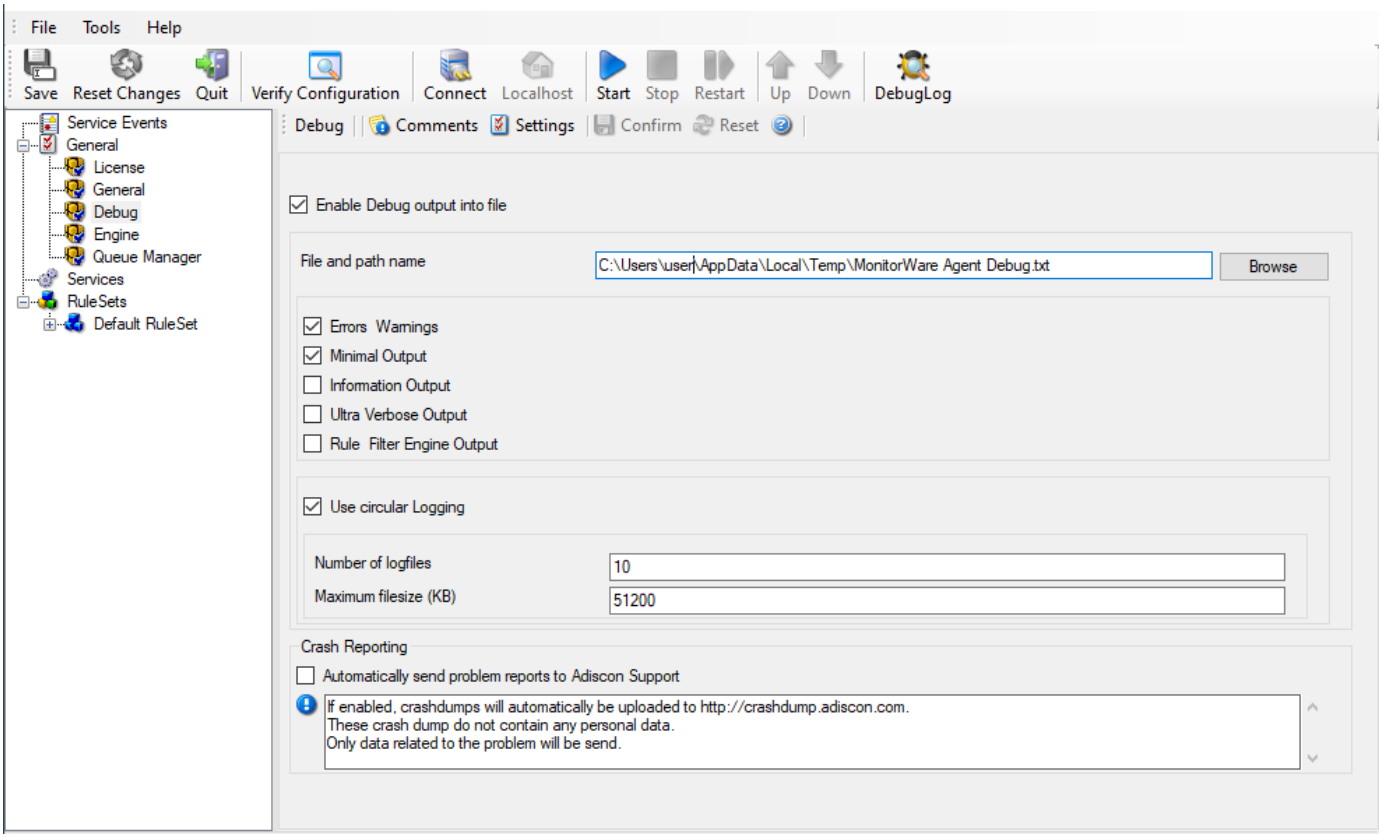
Debug

This tab can be used to debug rule bases. Especially with complex bases, it might be necessary to learn what application is internally doing while it is processing them. With the debug log, the service tells you some of these internal workings.

Other than rule basis testing, the debug log is also helpful when contacting Adiscon support. An Adiscon support engineer might ask you to set the debug log to a specific level while doing troubleshooting.

Note

Debug logging requires considerable system resources. The higher the log level, the more resources are needed. However, even the lowest level considerable slows down the service. As such, we highly recommend turning debug logging off for normal operations.



Enable Debug output into file

File Configuration field:

nEnableDebugOutput

Description

If checked, the debug log is enabled and written as the service operates. If unchecked, no debug log is written. For performance reasons, it is highly recommended that this box is unchecked during normal operations.

File and path name

File Configuration field:

szDebugFileName

Description

The full name of the log files to be written. Please be sure to specify a full path name including the drive letter. If just the file and/or path name is specified, that information is local to the service default directory. As this

depends on a number of parameters, it might be hard to find the actual log file. So for consistency purposes, be sure to specify a fully qualified file name including the drive.

Note: If the configured directories are missing, they are automatically created by application i.e. the folder specified in “File and Path Name”.

Debug Levels

File Configuration field:

nDebugErrors, nDebugMini, nDebugInternal, nDebugUltra, nDebugRuleEngine

Description

These checkboxes control the amount of debug information being written. We highly recommend only selecting “Errors & Warnings” as well as “Minimum Debug Output” unless otherwise instructed by Adiscon support.

Use circular Logging

File Configuration field:

nCircularLogging

Description

Support for circular debug logging has been added as the debuglog can increase and increase over time. This will avoid an accidental overload of the hard disk. Of course you can also customize the amount of files used and their size or disable this feature.

Automatically send problem reports to Adiscon Support

File Configuration field:

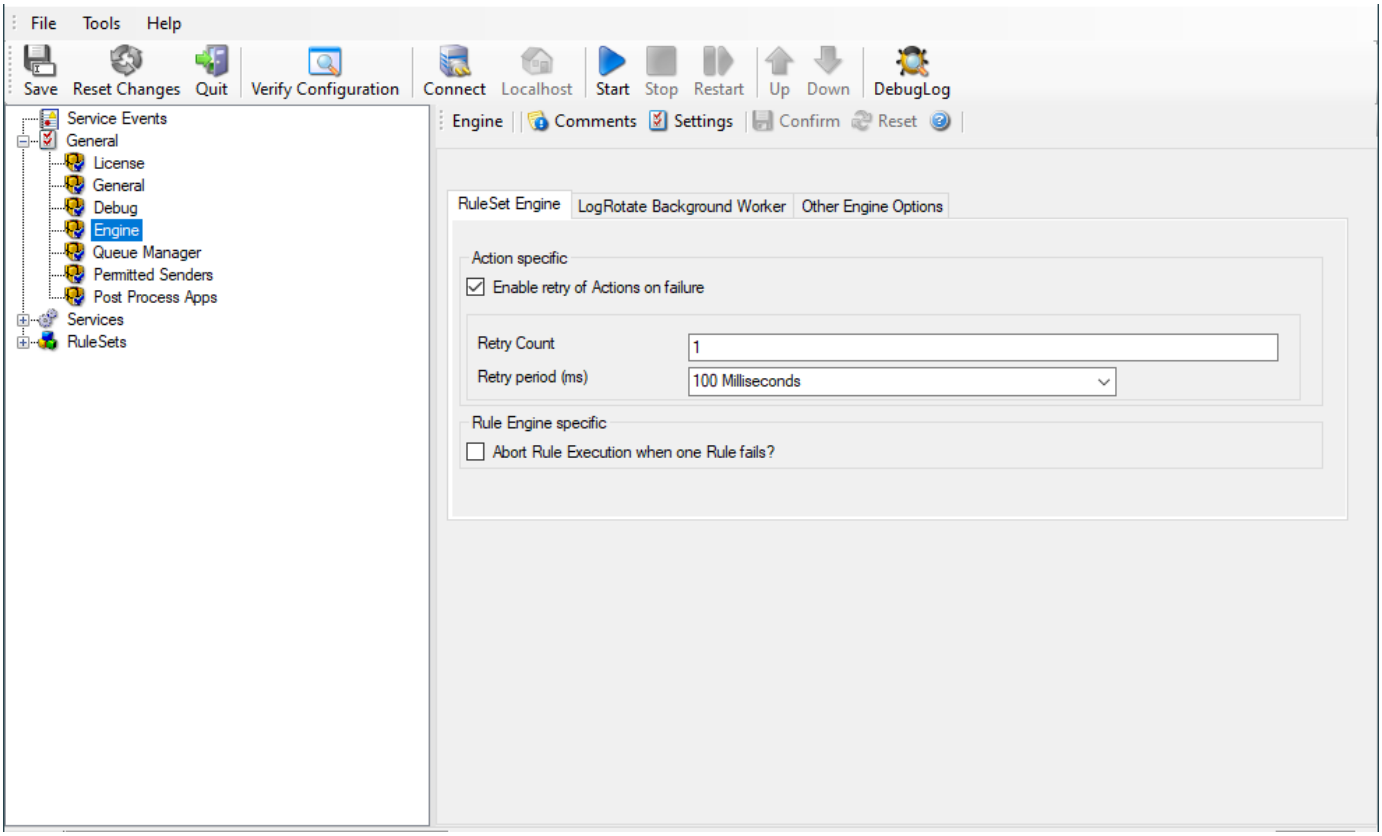
nReportCrash

Description

If enabled, problem reports will automatically be uploaded to `http://crashdump.adiscon.com`. A problem report is generated if the service internally stops working for some unknown reason. The reports are small dumpfiles which do not contain any personal data and will help us find and fix the problem. Also the dumpfiles are very small and do not exceed 256 Kbyte. In most cases only 32Kbyte data is send.

Engine

The Engine specific Options are explained below:



- RuleSet Engine Tab*

Action specific

Enable retry of Actions on failure

File Configuration field:

nEnableRetry

Description

If enabled, the Agent retries Actions on failure (until the retry counter is reached). Note that the Event error 114 will only be written if the last retry failed, previous error's will only be logged in the debug log (with the error facility). Note that you can customize the Retry Count and the Retry Period in ms as well.

Rule Engine specific

Abort Rule Execution when one Rule fails?

File Configuration field:

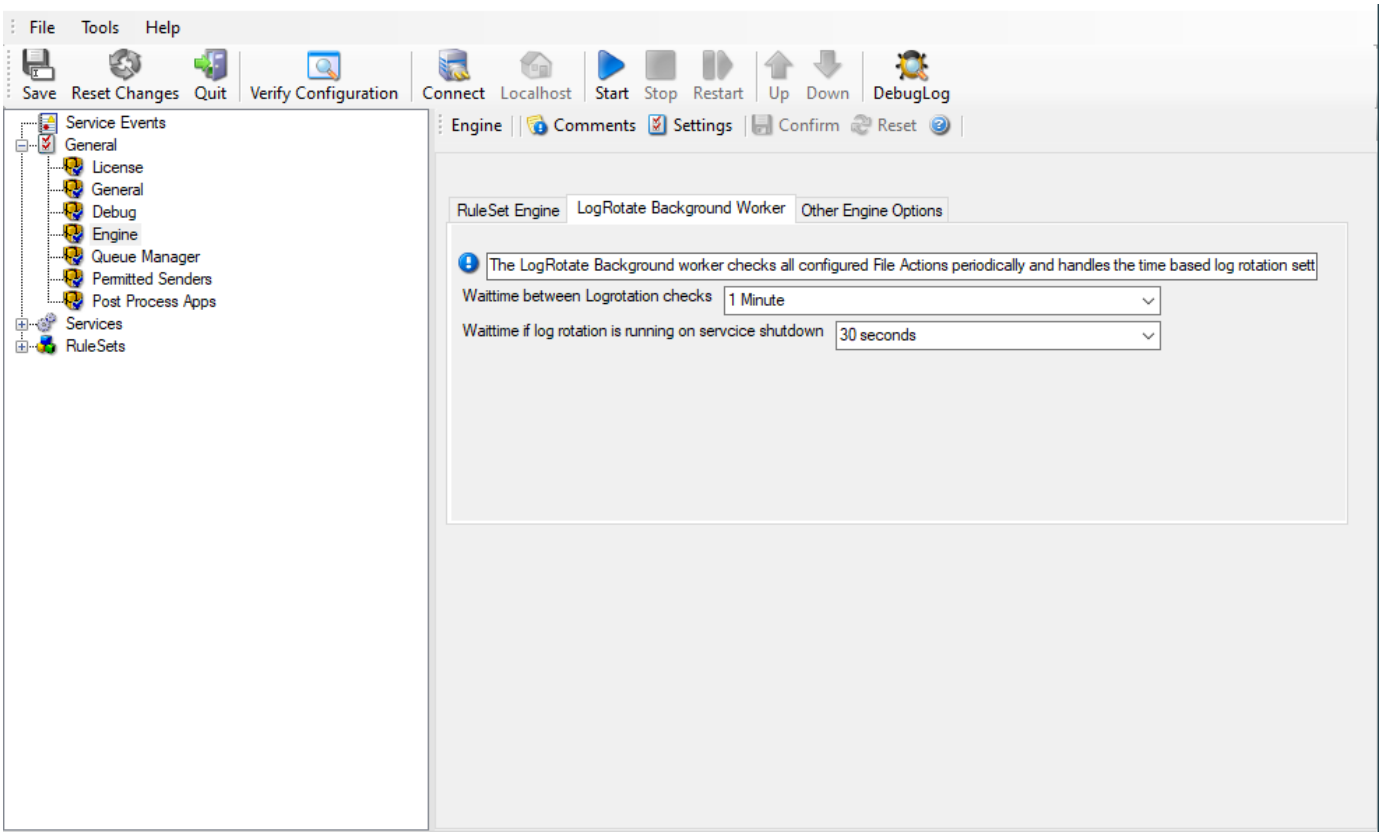
bAbortRuleOnFailure

Description

If checked, and an action fails, the execution will be aborted. If unchecked, and an action fails, simply the next action in this rule will be executed.

LogRotate Background Worker

The LogRotate Background worker checks all configured File Actions periodically and handles the time based log rotation settings, if enabled.



- LogRotate Background Worker Tab*

Wait time between Logrotation checks

File Configuration field:

nLogRotateWorkerSleepTime

Description

Defines how often the logrotate background worker thread checks all configured actions to see if any logfiles need to be rotated based on time related rotate conditions.

Wait time if log rotation is running on service shutdown

File Configuration field:

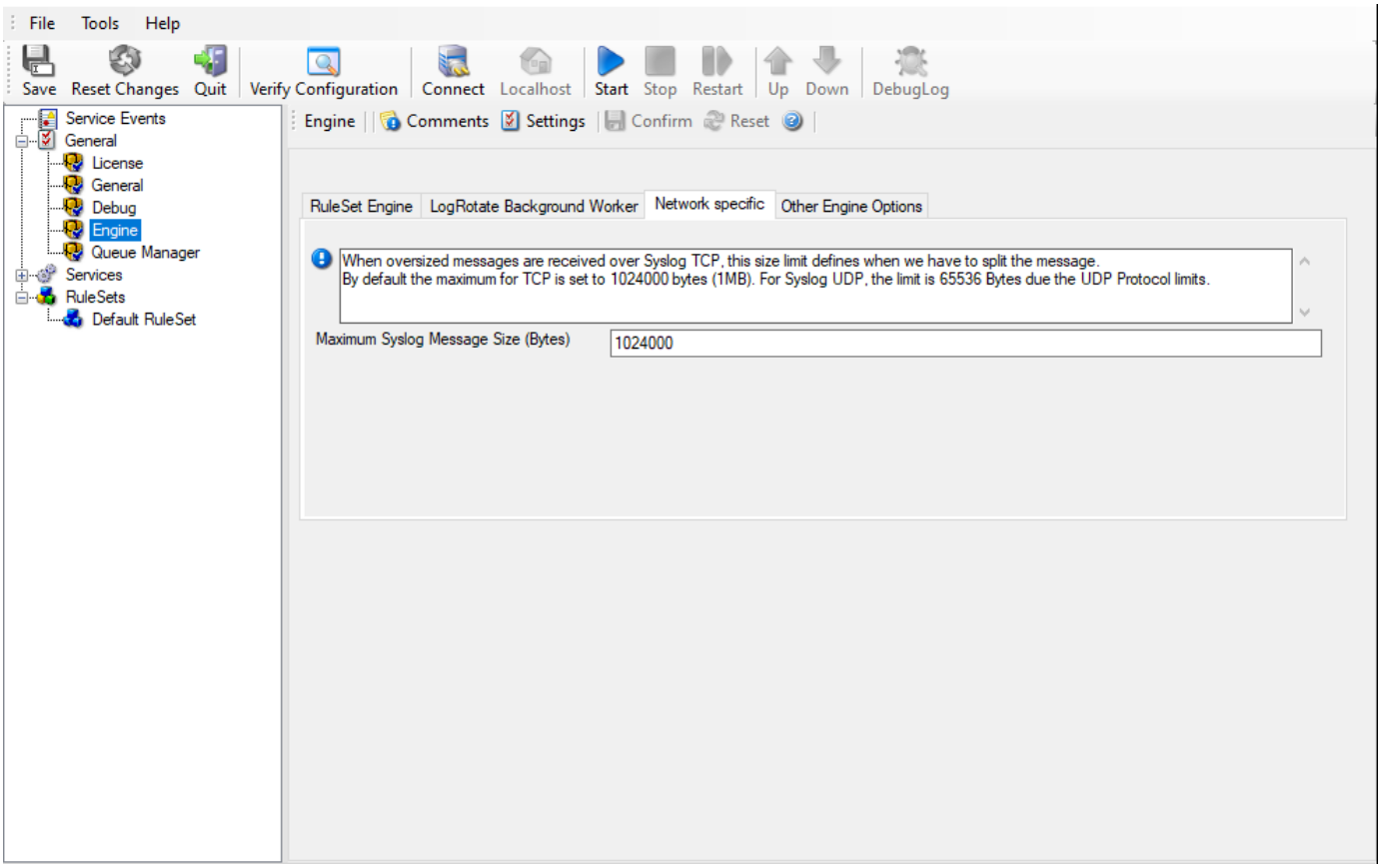
nLogRotateWorkerStopWaitTimeout

Description

When service is being shutdown, this defines how much time the logrotate background worker thread has left to finish its log rotations before the service exits.

If a queued or already active detached rotation does not finish within this bounded wait time, the unfinished detached work is saved and resumed on the next startup instead of extending shutdown indefinitely. For the full guarantee, see **`../shared/faq/log-rotation-guarantees`**.

Network specific Options



- Network specific Options Tab*

Maximum Syslog Message Size (Bytes)

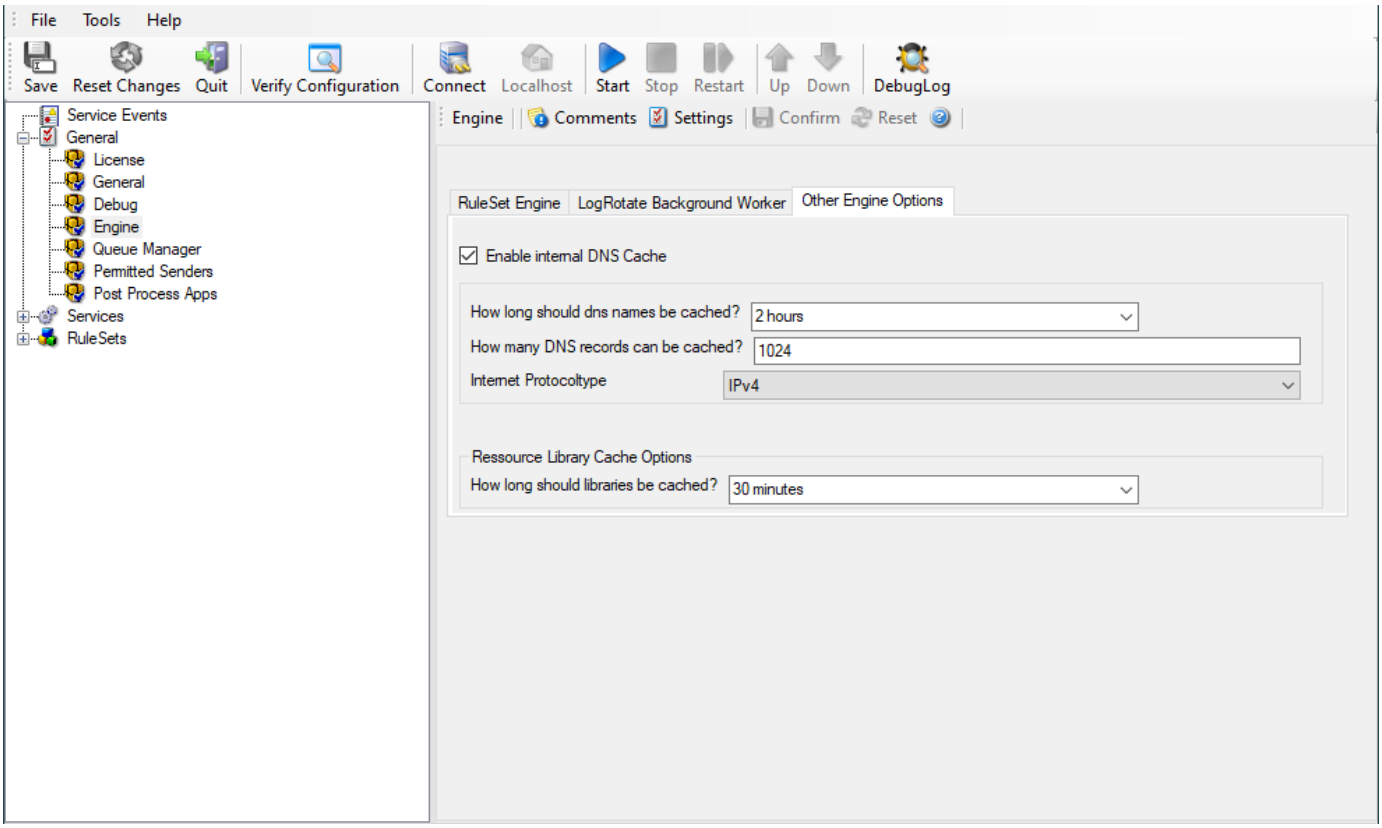
File Configuration field:

nSyslogMaxMessageSize

Description:

Configurable message size limit for Syslog TCP messages. The default is 1MB which is far more as defined in Syslog RFC's. If a syslog message exceeds the size limit, it will be split into multiple messages.

Other Engine Options



- Other Engine Options Tab*

Enable internal DNS Cache

File Configuration field:

nEnableDNSCache

Description

The DNS cache is used for reverse DNS lookups. A reverse lookup is used to translate an IP address into a computer name. This can be done via the resolve hostname action. For each lookup, DNS needs to be queried. This operation is somewhat costly (in terms of performance). Thus, lookup results are cached. Whenever a lookup needs to be performed, the system first checks if the result is already in the local cache. Only if not, the actual DNS query is performed and the result then stored to the cache. This greatly speeds up reverse host name lookups.

However, computer names and IP addresses can change. If they do, the owner updates DNS to reflect the change. If we would cache entries forever, the new name would never be known (because the entry would be in the cache and thus no DNS lookup would be done). To reduce this problem, cache records expire. Once expired, the record is considered to be non-existing in the cache and thus a new lookup is done.

Also, cache records take up system memory. If you have a very large number of senders who you need to resolve, more memory than you would like could be allocated to the cache. To solve this issue, a limit on the maximum number of cache records can be set. If that limit is hit, no new cache record is allocated. Instead, the least recently used record is overwritten with the newly requested one.

How long should DNS names be cached?

File Configuration field:

nDNSCacheTime

Description

This specifies the expiration time for cache records. Do not set it too high, as that could cause problems with changing names. A too low-limit results in more frequent DNS lookups. As a rule of thumb, the more static your

IP-to-hostname configuration is, the higher the expiration timeout can be. We suggest, though, not to use a timeout of more than 24 to 48 hours.

How many DNS records can be cached?

File Configuration field:

nDNSCacheLimit

Description

This is the maximum number of DNS records that can be cached. The system allocates only as many memory, as there are records required. So if you have a high limit but only few sending host names to resolve, the cache will remain small. However, if you have a very large number of host names to resolve, it might be useful to place an upper limit on the cache size. But this comes at the cost of more frequent DNS queries. You can calculate about 1 to 2 KBytes per cache record.

Internet Protocoltype

File Configuration field:

nDNSInetProtocol

Description

Select if you wish to prefer IPv4 or IPv6 addresses for name resolution. Note that this only has an effect on names which return both, IPv4 and IPv6 addresses.

Resource Library Cache Options

How long should libraries be cached?

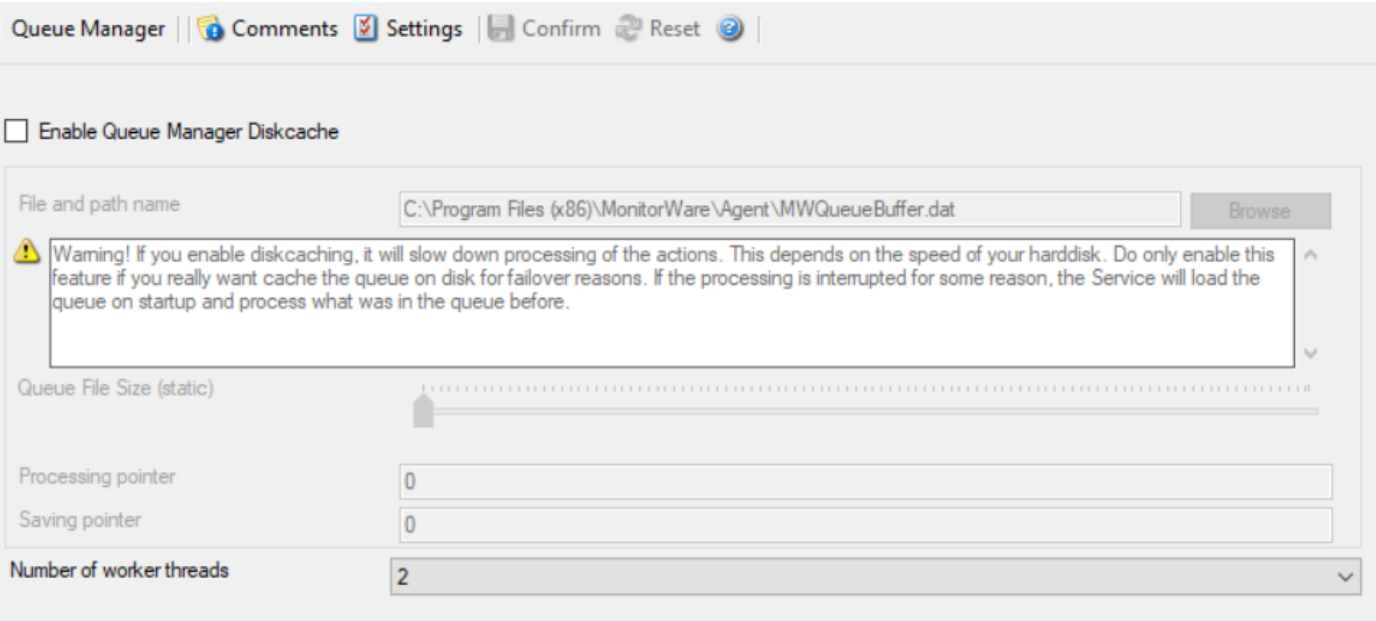
File Configuration field:

nLibCacheTimeOut

Description

This feature will be mainly useful for EventLog Monitor. For events with the same recurring event sources, this will be a great performance enhancement. The cache will also work for remote system libraries (requires administrative default shares). All libraries will be cached for 30 minutes by default.

Queue Manager



- Queue Manager*

Enable Queue Manager DiskCache

This feature enables the Agent to cache items in its internal queue on disk using a fixed data file.

> **Warning**
>
> Only use this feature if you really need to!

Depending on the speed of your hard disks, it will slow down processing of the actions, in worst case if the machine cannot handle the IO load, the Queue will become full sooner or later. The DiskCache is an additional feature for customers, who for example want to secure received Syslog messages which have not been processed yet.

The diskcache will not cache infounits from services like EventLog Monitor, as this kind of Service only continues if the actions were successfully. All other information sources like the Syslog server will cache its messages in this file. If the Service or Server crashes for some reason, the queue will be loaded automatically during next startup of the Agent. So messages which were in the queue will not be lost. Only the messages which was currently processed during the crash will be lost.

Enable Queue Manager Diskcache

File Configuration field:

nEnableRingBuffer

Description

Enable the disk based queue manager. Please read the description about the Queue Manager DiskCache first!

File and Pathname

File Configuration field:

szRingBufferFile

Description

As everywhere else, you can define here, where the queue file should be stored.

Queue File Size

File Configuration field:

nRingBufferSize

Description

With this slider, the queue size can be set from 1 MB to 2048 MB.

Processing pointer

File Configuration field:

nProcessingLow

Description

Points to the current processing position within the queue file.

Saving pointer

File Configuration field:

nSavingLow

Description

Points to the last processed position within the queue file.

Queue Manager specific

Number of worker threads

File Configuration field:

nWorkerThreads

Description

Defines the number of worker background threads that the core engine uses to process its queue.

Core concepts

Use this section to understand how EventReporter thinks about collected Windows events and how configuration objects interact.

Concept map

EventReporter processing follows this model:

1. An **input service** collects Windows Event Log data.
2. The collected data becomes an **information unit** inside EventReporter.
3. The **rule engine** evaluates the event against **rules** and **filter conditions**.
4. Matching **actions** store, forward, or transform the event.

In plain language, you can read this as:

```
input service -> ruleset -> action
```

Canonical concept pages

EventReporter services

Services inside EventReporter gather the Windows event data that is processed by rules. Each service type represents a collector with its own settings and behavior.

EventReporter is primarily built around Windows Event Log collection. In most installations, the active services are one or more Event Log Monitor instances.

A few key points matter:

- there can be multiple service instances as long as their settings do not conflict
- each service instance is bound to a ruleset
- service defaults are only templates, not active collectors
- if no service is configured, EventReporter does not collect any events

Information Units

Information units contain the data gathered by the services. As soon as a service detects a reportable event, it creates a new information unit. The information unit contains a textual representation of the event (for example a syslog message) as well as information about the event itself. For example, it contains the system that the event was originated from and the date and time it was received.

Which data is contained in the information unit depends on its type. However, there are a number of common data elements present in all information units. Most of these elements can be used as filter conditions in the rule engine. Information unit specific data elements are not eligible as filter conditions. However, there are data elements (properties) which are defined to be present in all information units even though they seem to be specific to a service type. One example is syslog priority. These values are present in each information unit type simply because priority is a good abstraction for other types, too. Such generally available properties are mapped if they are not directly supported by the service type. In the example, an Event Log Monitor maps the event log severity to the syslog priority.

There is a direct one-to-one relation between service type and information unit type. Each service type has its own information unit type.

Inside the rule base, the information unit type itself can be used as a filter condition. This facilitates creating rules that check information unit type specific properties only if they originated from the specific service type (e.g. check syslog priority only if the information unit was generated by a Syslog server).

Rules

Rules are the workhorse of the MonitorWare Agent. All actions and processing carried out is configured by the rules defined. Rules are configured by the client and processed by the so-called “rule engine” inside the MonitorWare Agent service.

You might already know something similar to the MonitorWare Agent rule engine. Rule engines and rule bases are an extremely powerful tool and in widespread use in the industry. Examples of rule bases can be found at Checkpoint’s Firewall One Firewall Rule Base or Cisco Routing filter - just to name a few.

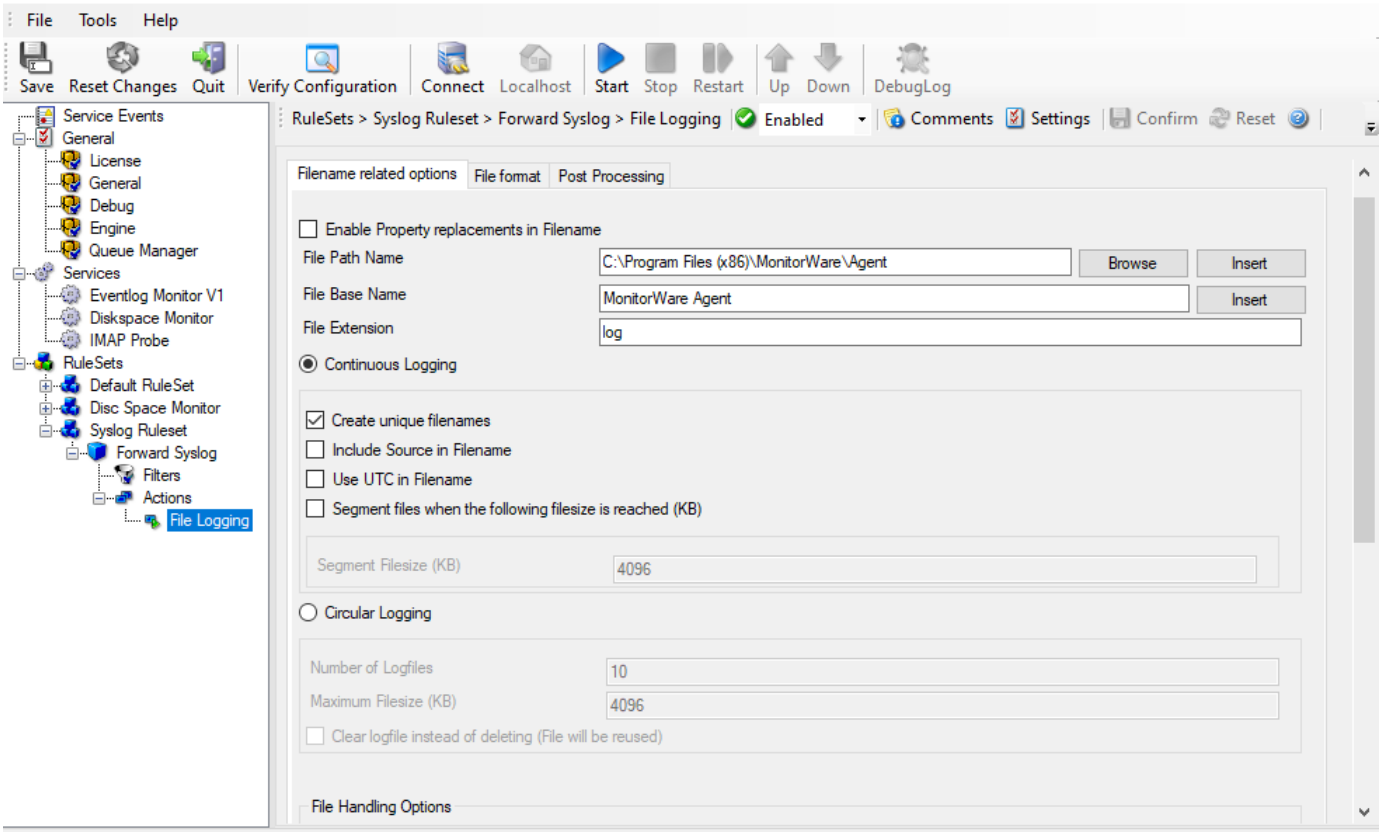
The rule base consists of the rules as configured in the client. The rule engine is the process carrying out the rules. A rule base can contain no, one or an unlimited number or rules. However, if there is no rule at all defined, no action will ever be carried out by the agent. Consequently, the client will issue a warning message in this case.

A rule has a description, associated match conditions, and actions. The match conditions are called “filter conditions”. These specify when a rule is to be carried out. Again, there can be no, one, or many filter conditions for a single rule. If there are no filter conditions, the rule will always match. This is useful in many cases. If there is more than one filter condition, all filter conditions need to match in order for the rule to match (logical AND).

Actions associated with a rule specify what to do when the associated rule matches (and only the associated rule). Actions carry out the actual processing of messages. For example, actions include logging a message to a flat file or database, sending it via email or forwarding it to syslog daemon or another MonitorWare Agent. There can be no, one, or an unlimited number of actions associated with a rule. However, if no action is associated, the rule will not have any effect. Consequently, the client will issue a warning when writing the rule base. Rules without actions can be useful to temporarily disable a rule with complex filter condition. If there are multiple actions, they are not guaranteed to be carried out in any specific order. If you definitely need an action to be carried out before another one, you currently need to define two rules.

Actions can be modified with action modifiers. These are the strings attached to a specific action. Action modifiers allow customizing a specific behavior of this action. It modifies only this action and only this one, other actions of the same type are not affected - regardless if the appear in the same rule or a very different one. The use of the action modifier depends on the type of action. For example, with syslog forwarding it is the host the syslog message is to be forwarded to. With ODBC database logging it is the DSN and so on. If there is no action modifier, the values configured in the client’s configuration tabs will be used. They are also used for all values that cannot be modified via the action modifier (e.g. the SMTP server address for email forwarding).

Below find a screenshot of a rule base with a number of rules, filter conditions and action modifiers:



Sample Rule Base

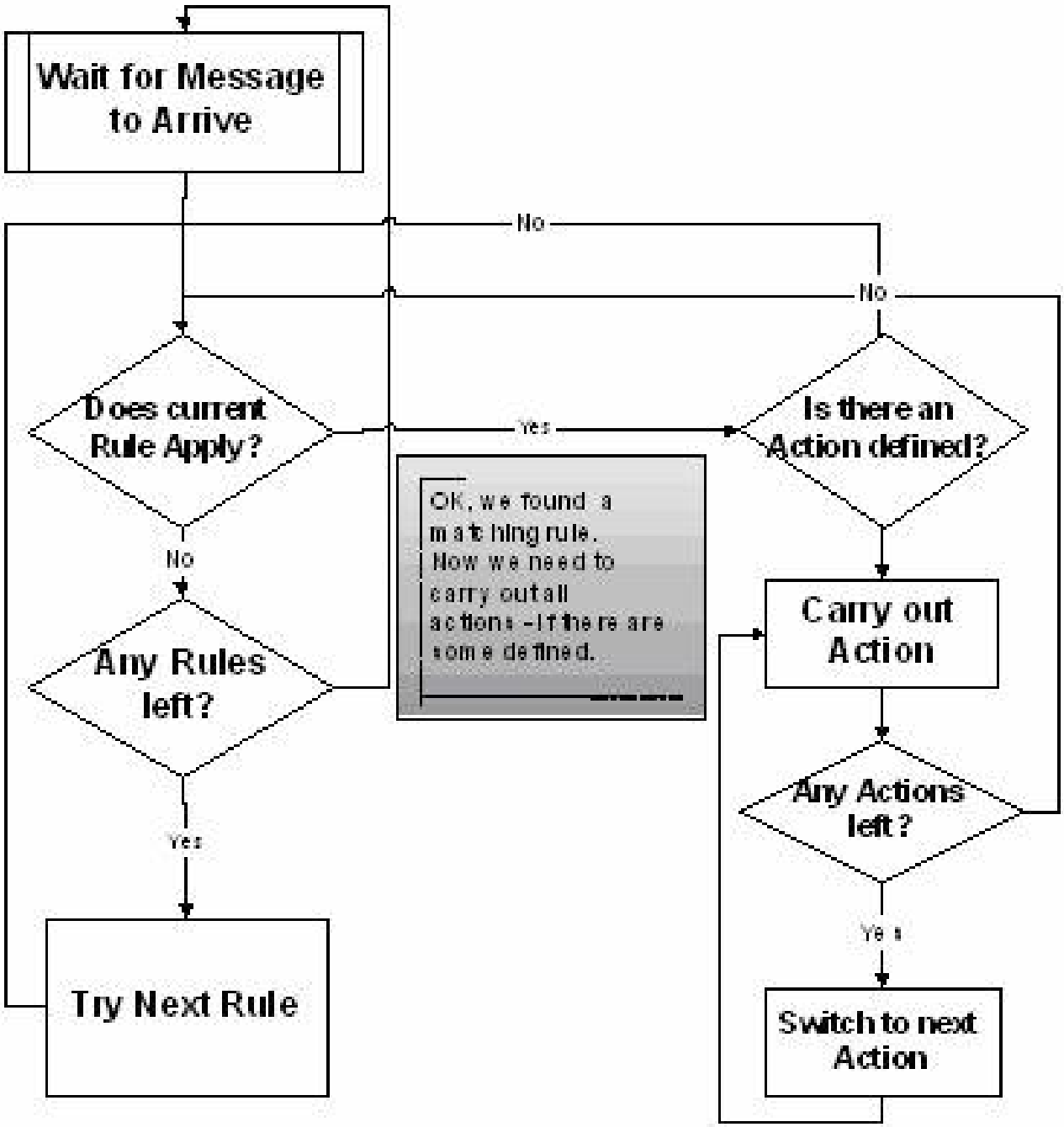
Now that we know the elements, how are rules being processed. It is easy. Rules are strictly processed from top to bottom, or from number one to the last one. For each rule the filter conditions are checked to see if they match. If they do, all associated actions are carried out. Then, the rule engine advances to the next configured rule. Once again, it checks if it matches and - if it does - carries out the actions associated with that rule. Then the next rule is processed and so on. The rule engine stops when there are no more rules to be evaluated. It also stops if a rule contains a “discard” action.

The “discard action” is a very special and powerful action. It does not actually carry out any processing. In fact, it disables all further processing for a message as soon as it is found by the rule engine. So what is the discard action good for? It is used to handle common situations where a number of well know messages - unimportant messages - should be filtered out so that the other rules do not need to take care of these messages. In many other products using rules bases, this is called the “block rule”. Please note that with Adiscon’s rule engine, there can be multiple block rules at multiple layers of the rule base giving you additional flexibility.

One last thing to mention: the rule base is applied to every message arriving at the MonitorWare Agent. By design, there is no way to modify the behavior of the rule base for the next message to be arrived. This ensures an always consistent processing of incoming messages. However, there can be multiple rule bases. Each rule base is associated with a service. Only the rule base associated with the service generating the message will be processed.

While building and testing your rule base, please keep in mind that the MonitorWare Agent service needs to be restarted to load a modified rule base. The reason is that the service does not re-read the rule base to save system resources.

There is an online seminar available on the rule engine and its processing. If you are interested in a more in-depth view, you might want to visit it at rule engine.



Rule Engine Flowchart

For those interested in more in-depth information on how the rule engine works, this flowchart might be helpful:

There is an online seminar available on the rule engine and its processing. If you are interested in a more in-depth view, you might want to visit it at rule engine.

The Rule Engine

Overview

This paper explains you the Rule Engine that is employed in some of the MonitorWare Line of Products namely MonitorWare Agent, WinSyslog, and EventReporter 6.0 (and higher)

What is the Rule Engine

Rule Engine is actually an engine present in the above mentioned MonitorWare Line of Products using which you can define certain filters and the actions that are to be carried out if the defined filter condition matches with the real time condition.

Rule Engine revolves around four basic concepts:

- Information Unit
- Information Services
- RuleSets
- Queue Manager

In order to understand the complete Rule Engine, you need to understand the above mentioned four concepts. The details of these are written below

1. Information Unit (Info Unit)

“Information Unit” or “Info Unit”, as we call it, is the basic building block of Rule Engine. Info Unit is basically an object that contains all the information about a specific event which includes:

- Message
- Which application generated this event
- When this event was generated
- Syslog Facility
- Syslog Priority
- Info Unit Type (it tells which Info Service has generated this Info Unit)
- etc

The following figure will give you an idea about an Info Unit:

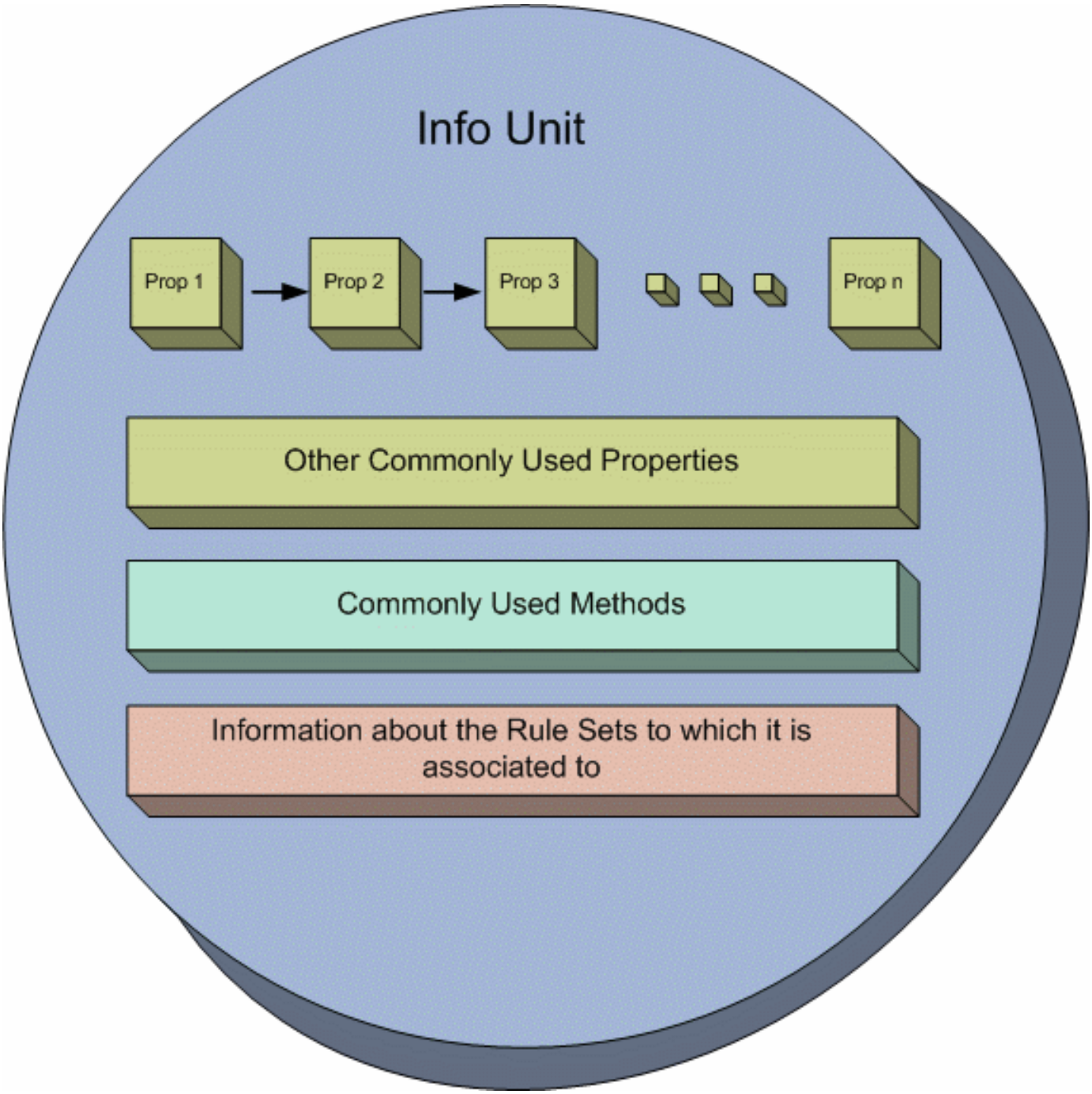


Figure 1: Conceptual Diagram of an Info Unit

As the figure illustrates, an Info Unit contains most of the properties (mentioned above in bullets) in the form of a list. In addition to this list, it also has some commonly used properties separately stored in it (for efficiency reasons). Apart from the properties, an Info Unit also has some methods which allow it to write it or to construct itself etc. Information about the RuleSets (will be explained in the coming sections) is also contained within each Info Unit so that it exactly knows which rules will be applied on it.

2. Information Services (Info Service)

“Information Services” or “Info Services”, as we call them, generate Info Units. Each Information Service will generate its own Info Units. The important thing to note over here is that each Info Unit has the same format but can have different properties and rulesets associated with it. For example, if an architect makes a building plan then it becomes a template. Now he can use this template to construct as many buildings as he likes but each one can have different properties (they can differ in color scheme, window styles etc). Exactly in the similar way, an Info Unit is actually a template from which each Info Service makes a specific object of Info Unit that might differ in properties from another Info Unit object.

Examples of Info Services

There can be a number of different examples on Info Services. Following are some of the examples:

1. Syslog server

It receives the messages that are forwarded to it and for each message (or event) it generates an Info Unit out of it.

2. Event Log Monitor

It picks up the events from the Window's Event Log and for each event it constructs an Info Unit.

3. Ping Probe

It pings a specified device and if doesn't find a response from the other side, it generates an Info Unit with desired information.

Important Note

One thing to note about Info Services is that there can a number of different Services running on the same machine. You can even run the different instances of the same Info Service (but with different properties naturally). In either case, each Info Service will generate its own Info Unit.

3. RuleSets

As the name suggests, a RuleSet is a set of Rules. A "Rule" consists of the following two things

- Filter Condition
- Actions

You might have noticed that the point 1 written above is singular and point 2 is plural which clearly means that you can define only one Filter condition for one rule but can define as many actions as you like. The filter condition can however contain as many Boolean operators as you like.

Filter Condition

Filter Condition is a combination of different Boolean operators which will evaluate to a Boolean answer. In simple words, the result of a filter condition can either be True or False.

Actions

Actions are all those events which are fired when a filter condition evaluates to a True value. As mentioned above, a Rule can have more than one actions associated with it which means that if a filter condition evaluates to a true value then all of the actions associated with that rule will execute. If the filter condition, on the other hand, evaluates to a false value then all of the defined actions will be skipped.

Note that other than normal actions, there are three special kinds of Actions that are worth mentioning here:

- Discard Action (Explained Later)
- Include Action (Explained Later)
- Actions that can alter the contents of Info Units permanently

4. Queue Manager

Queue Manager simply maintains a queue of all of the Info Units that have been forwarded to it by different Info Services.

Overall Picture

This section will explain you that how the different components are related to each other and how does the whole process work. The picture shown below gives an idea about how things are working. As you can see that we have four different stages through which the events are processed.

Info Services picks up the events and convert them into Info Unit. Note that each Info Service has its own Info Unit. These Info Units are passed to the Queue Manager. The job of the Queue Manager, as mentioned above and as clear from the diagram, is to simply make a queue of these Info Units that it has received from various Info Services. The Rule Engine picks up the Info Units from this Queue Manager, applies the rules on these Info Units (as mentioned above, each Info Unit has the information about which rules should be applied on it) and if necessary

carries on the actions. The rule engine keeps on repeating this process while there are some Info Units present in the Queue.

Manager's Queue

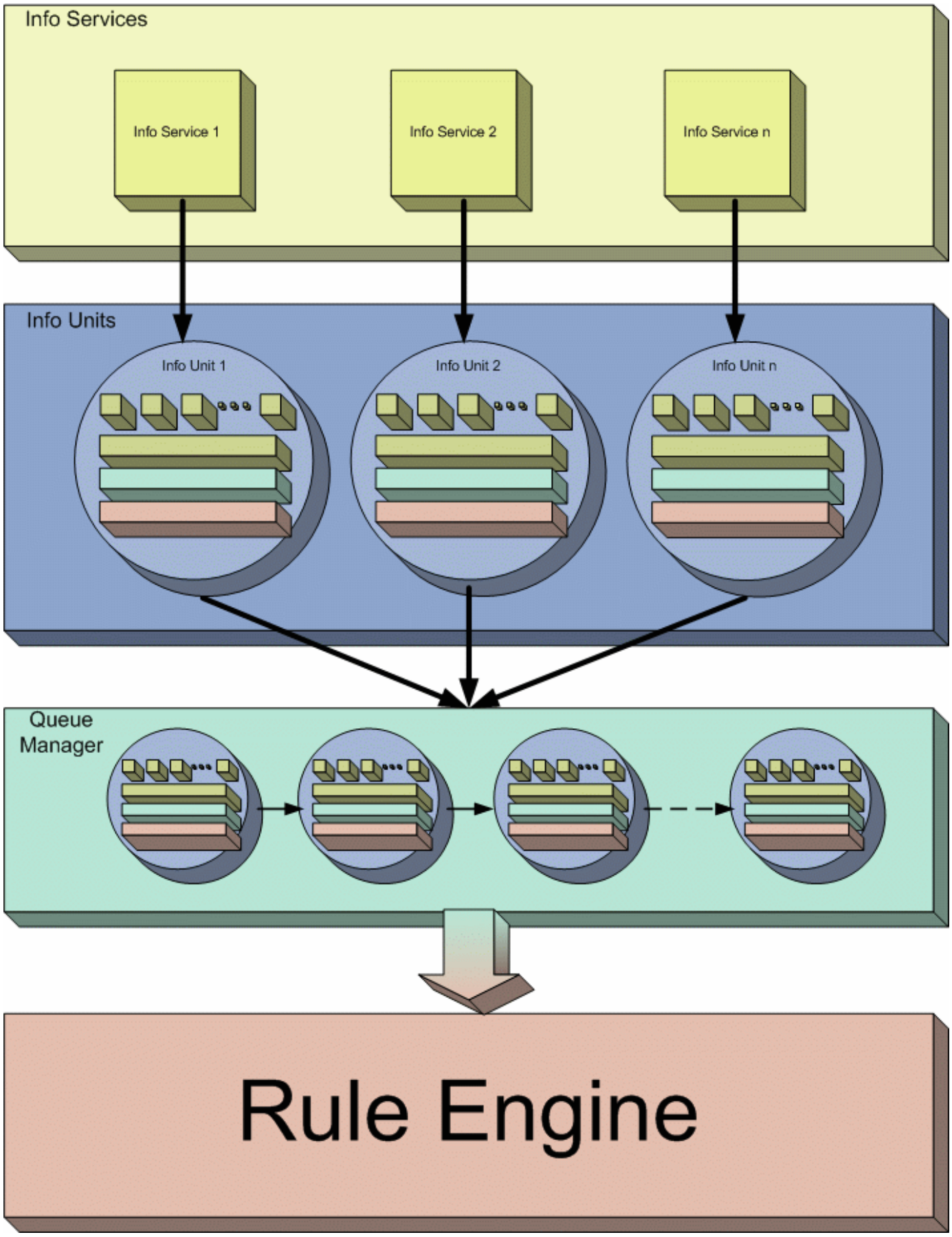


Figure 2: Overall Process

How Does the Rule Engine Work

Having explained the overall picture of the whole process, let’s specifically talk about Rule Engine. The following figure explains it in detail:

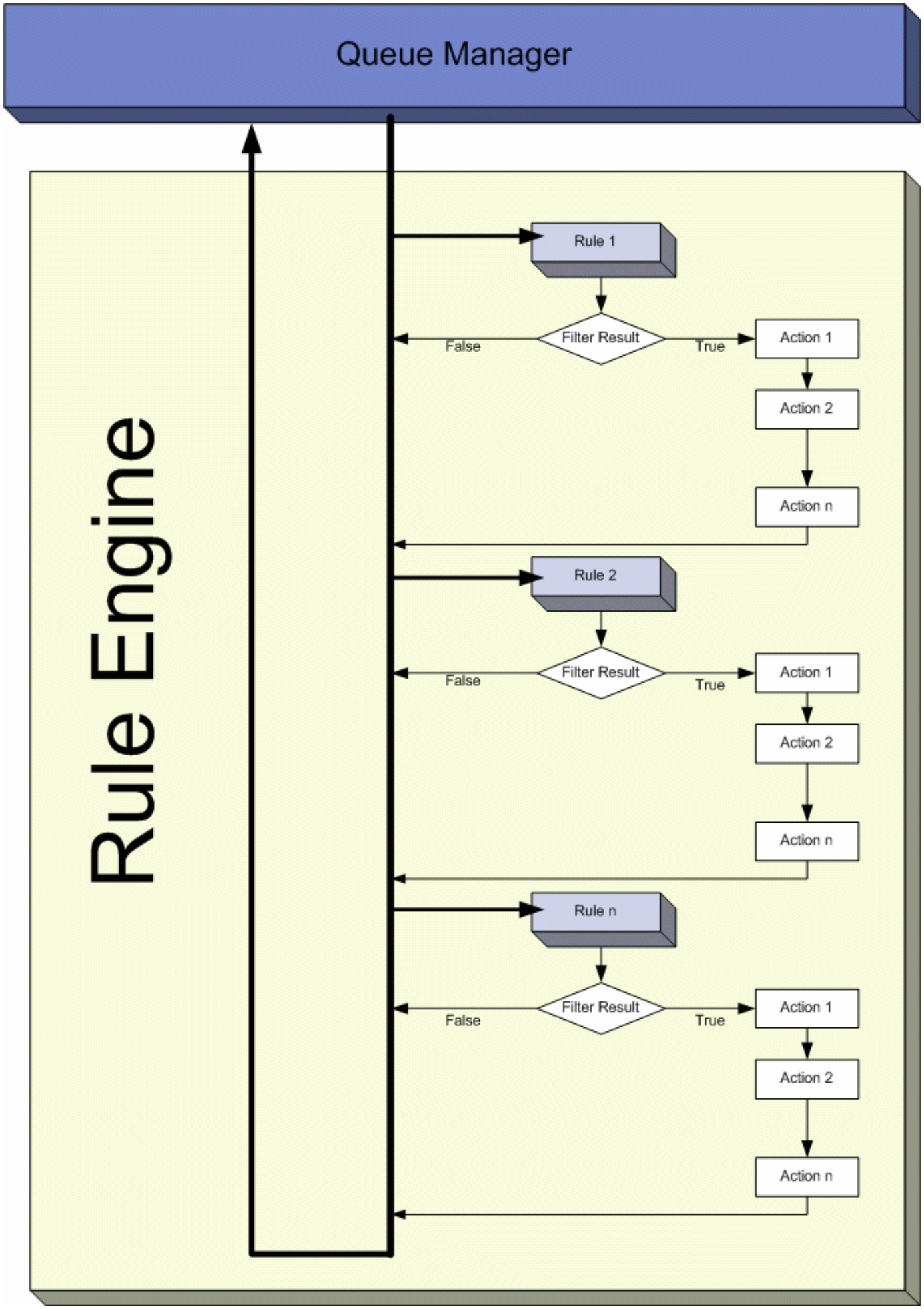

Queue Manager
Rule Engine
Rule 1
Filter Result
False
True
Action 1
Action 2
Action n
Rule 2
Filter Result
False
True
Action 1
Action 2
Action n
Rule n
Filter Result
False
True
Action 1
Action 2
Action n

Figure 3: Working of Rule Engine

As you can see in the figure, the Rule Engine picks up Info Units one by one from the Queue Manager. Since each Info Unit has the information about its Rule sets, it will apply the rules on it in the same order in which they were defined. As you can see above, it will pick up the first Rule and evaluates its Filter Condition. If that Filter Condition is evaluated to false, all the actions associated with that rule will be skipped and it will pick up the second Rule. If the Filter Condition, on the other hand, evaluates to True, it will execute all the actions that are associated with this Rule in the order in which they were defined. After the execution of all these actions, it will pick up the next rule in the current ruleset. Once all the rules have been executed, the current Info Unit (that was handed over to Rule Engine by the Queue Manager) will be destroyed and the Rule Engine will go to the Queue Manager to pick up the next Info Unit if there exists one.

The above picture has been drawn for normal flow of executions. There can be 2 conditions when the flow will not follow the diagram shown above. These conditions arise in response to 2 special kinds of actions that are called Discard Action and Include Action.

Discard Action

A Discard Action immediately destroys the current Info Unit and any action of any Rule that has been defined after the Discard Action will not be executed at all. Let's take a simple example to clarify it further.

Let's say that Action 2 of Rule 1 in the picture above is a Discard Action. If the Filter Result of Rule 1 is evaluated to true, then Action 1 will be executed. As Action 2 is a Discard Action, immediately the current Info Unit will be destroyed (which means that now the Rule Engine will skip all the Rules and all the actions associated with them) and the Rule Engine will go back to the Queue Manager to pick up the next Info Unit in the Queue.

Include Action

An Include Action simply includes another RuleSet in some existing RuleSet. When this Action is encountered, the Rule Engine leaves the normal flow and go to the included ruleset (which may contain many rules as well). It executes all the rules that have been defined in that included RuleSet. After the execution of all of them, it will return to its point from where it left the original flow. Let's take an example to clarify it a little further.

Let's say that the Action 1 or Rule 1 is an include action. If the Filter Condition result of Rule 1 evaluates to true, it will execute the Action 1. Since Action 1 is the include action in this example, it will go to the included ruleset and will execute its Filter Condition. If that filter condition evaluates to true, it will execute all of its actions and will return to Action 2 of Rule 1 (of normal flow) and if on the other hand, the filter condition of the included ruleset evaluates to false, it will skip all of its actions and will come back to the Action 2 of Rule 1 (of normal flow).

Note that there is no limit on including the rules which means that a rule that has been included in another rule may contain another rule in it which might contain another rule in it and so on.

Suggestions for Defining Complex RuleSets

While defining a complex RuleSet, it might be a good idea to follow the stages defined below.

Edit Stage # - Actions Stage 0 - Discard unwanted events Stage 1 - Post Process Stage 2 - Discard unwanted events Stage 3 - All Actions Stage 4 - Individual Actions

As mentioned above, the rules and actions will be executed in the order in which you will define them. So it's very important that you define the actions in a way such that you achieve the desired results as well as achieve them with efficiency. For example, if you haven't defined any filter which we call as No Filter (it always evaluates to true) and if the first action that you have defined is the Discard Action, then there is no meaning of defining any action after this first action because the first action will always be executed and it will always discard the complete Info Unit.

Here is the explanation of the above mentioned stages.

Stage 0

In this stage, you can discard those events that you are not interested in. You can use the Discard Action explained above to discard the events.

Stage 1

In this stage, we recommend to Post Process the incoming Info Units. Once the Info Unit has been handed over to the Rule Engine from the Queue Manager, you can actually change the contents of the Info Units to make them more meaningful.

Stage 2

In this stage, you might want to again discard those events that you are not interested in. Simply use the Discard Action.

Stage 3

In this stage, you will apply the actions that will apply to all of the Info Units coming (to be more specific, you will apply those rules over here for which you have selected “No Filter” as the filter condition.

Stage 4

In this stage, you will create the rules for which you have specific filter conditions.

To sum it up, we recommend doing most generic things first and least generic things later or in other words, do the generic things first and the specific things later. Note that this section suggests only the typical scenario but it can vary from depending upon the needs. For example, you might want to perform some actions on some specific events after stage 1 and before stage 2.

EventReporter filter conditions

Filter conditions define when a rule should match a collected Windows event. They are the main way to separate high-value events from background noise.

A filter can be simple, such as one event ID check, or complex, with nested Boolean logic across multiple properties.

A few key points matter:

- rule matching depends on the full filter expression
- an empty top-level condition matches everything
- filter order inside the Boolean tree affects the result
- precise filtering is usually easier if you first verify the broad event path

Actions

Actions tell the Product (i.e. MonitorWare Agent or EventReporter or WinSyslog or any of the combinations) what to do with a given event. With actions, you can forward events to a mail recipient or Syslog server, store it in a file or database or do many other things with it. There can be multiple actions for each rule. These actions are described in the following section.

Write to File

The message is written to a plain text log file.

Write to Database

The message will be written to the specified ODBC database. This database format will be used by the MonitorWare Console that becomes available later. Therefore, if you intend to use the console, we recommend adding at least one rule that persists data to the database.

Write to EventLog

The message will be written to the application event log. Please note that the agent intentionally does not try to make the message look like it was generated on the local system. This could be very confusing. Instead, it is written inside the message part with standard values for event source and type.

Forward via Email

The message will be forwarded via email. Please note that each message will generate one email message. Messages are not combined to fit into a single mail. The Send Mail Action includes a timeout feature (m_nTimeoutValue) that provides control over message delivery timing.

Forward via Syslog

The message will be forwarded to a syslog daemon. UDP and TCP forwarding is supported.

Forward via SETP

The message will be forwarded via the custom SETP protocol. This is typically used in environments where data from different agents will be consolidated in a central place. SETP allows to transfer all InformationUnits exactly as they are. As such, the central repository can store an exact picture of the whole network.

Net Send

The message will be forwarded via the Windows “net send” functionality. Please note that the Windows function is not very reliable and requires the user to be logged in. As such, we recommend using “Net Send” only in combination with other actions.

Start Program

The message will be passed to an external process. The command line is specified in the action modifier.

Play Sound Action

This action allows you to play a sound file.

Send to Communications Port

This action allows you to send a string to an attached communication device, that is it sends a message through a Serial Port. It can send any message to a configured Serial or Printer port.

Set Status

This action allows you to create new properties of your own choice in the incoming messages. There is an internal Status List within the product which you can use for more complex filtering. You can set property over the Set Status action and you can add filter for them. They are more or less helpers for building complex rule constructions.

Set Property

With the “Set Property” action, some properties of the incoming message can be modified. This is especially useful if an administrator would like to e.g. rename two equally named devices.

Call RuleSet

This Action simply calls another RuleSet in some existing RuleSet. When this Action is encountered, the Rule Engine leaves the normal flow and go to the called RuleSet (which may contain many rules as well). It executes all the rules that have been defined in that called RuleSet. After the execution of all of them, it will return to its point from where it left the original flow.

Discard

Please see the rules description below for a complete discussion. Effectively, the message will be discarded and any further processing of this information unit be stopped as soon as a “Discard” action is found.

Post-Process Event Action

The post process action allows you to re-parse a message after it has been processed e.g. Tab Delimited format. Such re-parsing is useful if you either have a non-standard syslog format or if you would like to extract specific properties from the message.

Why this matters

Understanding these concepts helps you:

- design rulesets with predictable behavior for each input service
- avoid duplicate or conflicting processing paths
- choose the right action type for storage, forwarding, or alerting
- troubleshoot why an event did or did not match a rule

Services

Services are the configured EventReporter inputs. They collect Windows event data and pass it to rulesets for further processing.

In EventReporter, the most important service types are the Windows Event Log monitor services. They read Windows events and make them available to the rule engine.

In this manual, **input** is the clearest plain-language concept, while **service** remains the operational term for the configured EventReporter object.

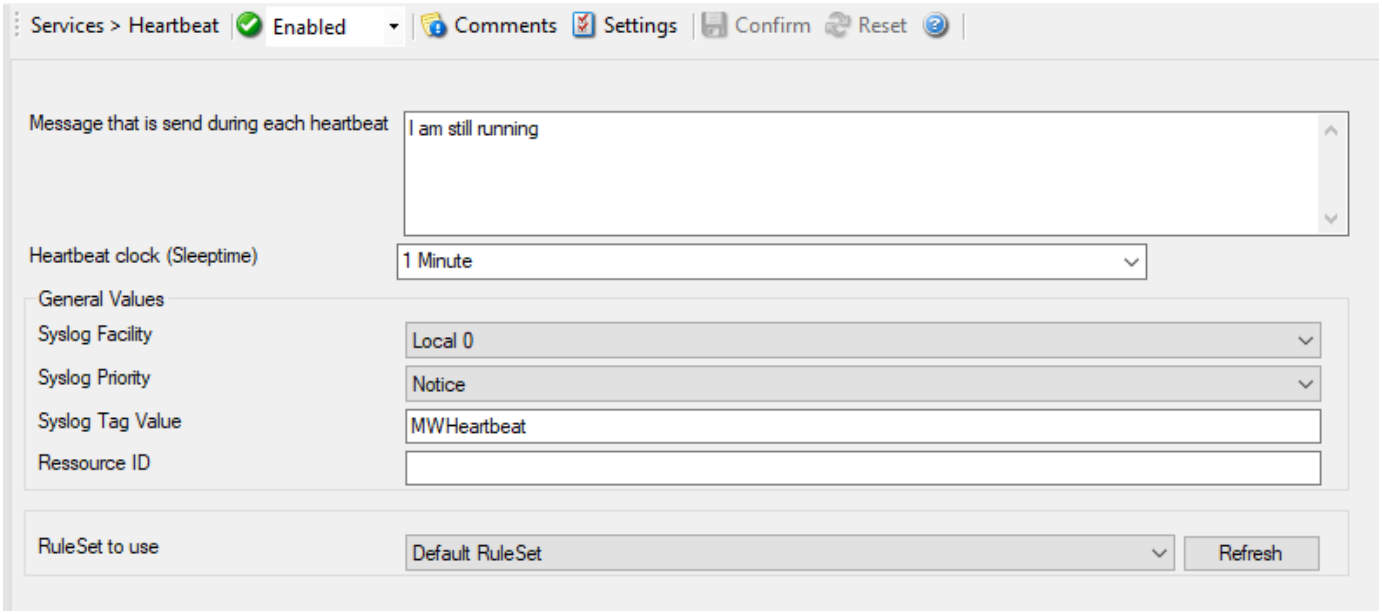
Important behavior

- You must define at least one input service, otherwise EventReporter has no event data to process.
- A service instance is an active collector.
- A service default is only a template. It does not collect anything until you create an actual service instance from it.
- Each service instance is bound to one ruleset.

Service types used in EventReporter

Heartbeat

The heartbeat process can be used to continuously check if everything is running well. It generates an information unit every specified time interval. That information unit can be forward to a different system. If it does not receive additional packets within the configured interval, it can be assumed that the sender is either in trouble or already stopped running.



- Service - Heartbeat*

Message that is send during each heartbeat

File Configuration field:

szMessage

Description:

This is the message that is used as text inside the information unit. Use whatever value is appropriate. The message text does not have any special meaning, so use whatever value you seem fit.

Heartbeat clock (Sleeptime)

File Configuration field:

nSleepTime

Description:

This is the interval, in **`milliseconds`**, that the heartbeat service generates information units in. Please note that the receiving side should be tolerant. The interval specified here is the minimum time between packets. Under heavy load, the interval might be slightly longer. It is good practice to allow twice this interval before the service is considered suspect by the system monitoring the services health.

General Values (Common settings for most services)

Syslog Facility

File Configuration field:

nSyslogFacility

Description:

The syslog facility to be assigned to events created by this service. Most useful if the message is to forward to a Syslog server.

Syslog Priority

File Configuration field:

nSyslogPriority

Description:

The Syslog priority to be assigned to events created by this service. Most useful if the message is to forward to a Syslog server.

Syslog Tag Value

File Configuration field:

szSyslogTagValue

Description:

The Syslog tag value to be assigned to events created by this service. Most useful if the message is to forward to a Syslog server.

Resource ID

File Configuration field:

szResource

Description:

The resource id to be assigned to events created by this service. Most useful if the message is to forward to a Syslog server.

RuleSet to Use

File Configuration field:

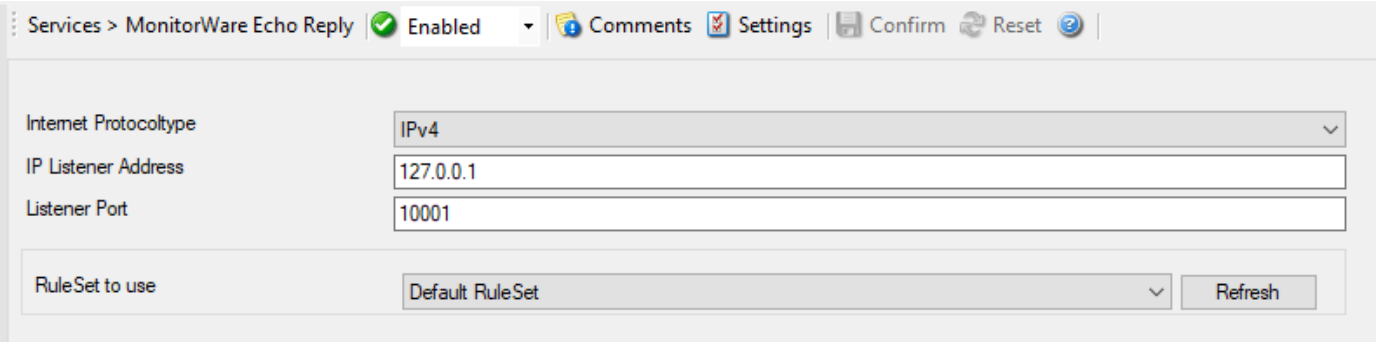
szRuleSetName

Description:

Name of the ruleset to be used for this service. The RuleSet name must be a valid RuleSet.

MonitorWare Echo Reply

The Echo Reply service is used on each of the installed EventReporter/ MonitorWare Agent. A central agent running the MonitorWare Agent is using the echo request and instructs to poll each of the other EventReporter/MonitorWare Agent services. When the request is not carried out successfully, an alert is generated. The MonitorWare echo protocol ensures that always a fresh probe of the remote EventReporter/MonitorWare Agent Service is done.



- Service - MonitorWare Echo Reply*

Internet Protocoltype

File Configuration field:

nInetType

Description:

Select the desired protocol type. IPv4 and IPv6 are available. The IPv6 protocol needs to be properly installed in order to be used. Note that one Service can only handle IPv4 or IPv6, so if you want to use both protocols, you will need to create two separate services.

IP Listener Address

File Configuration field:

szMyIPAddress

Description:

The MonitorWare Echo Reply service can be bound to a specific IP Address. You can either use an IPv4, an IPv6 Address, or a Hostname that resolves to an IPv4 or IPv6 Address. This feature is useful for multihome environments where you want to run different Syslog Servers on different IP Addresses. Please note that the default IP Address `0.0.0.0` means ANY IP Address.

Listener Port

File Configuration field:

nListenPort

Description:

Specify the listener port here.

RuleSet to Use

File Configuration field:

szRuleSetName

Description:

Name of the ruleset to be used for this service. The RuleSet name must be a valid RuleSet.

Event Log Monitor V1

This dialog configures the Windows Event Log Monitor service.

This service was initially introduced by **`Adiscon's EventReporter`** product. To allow previous EventReporter customers seamless upgrades, there are a number of compatibility settings to support older message formats.

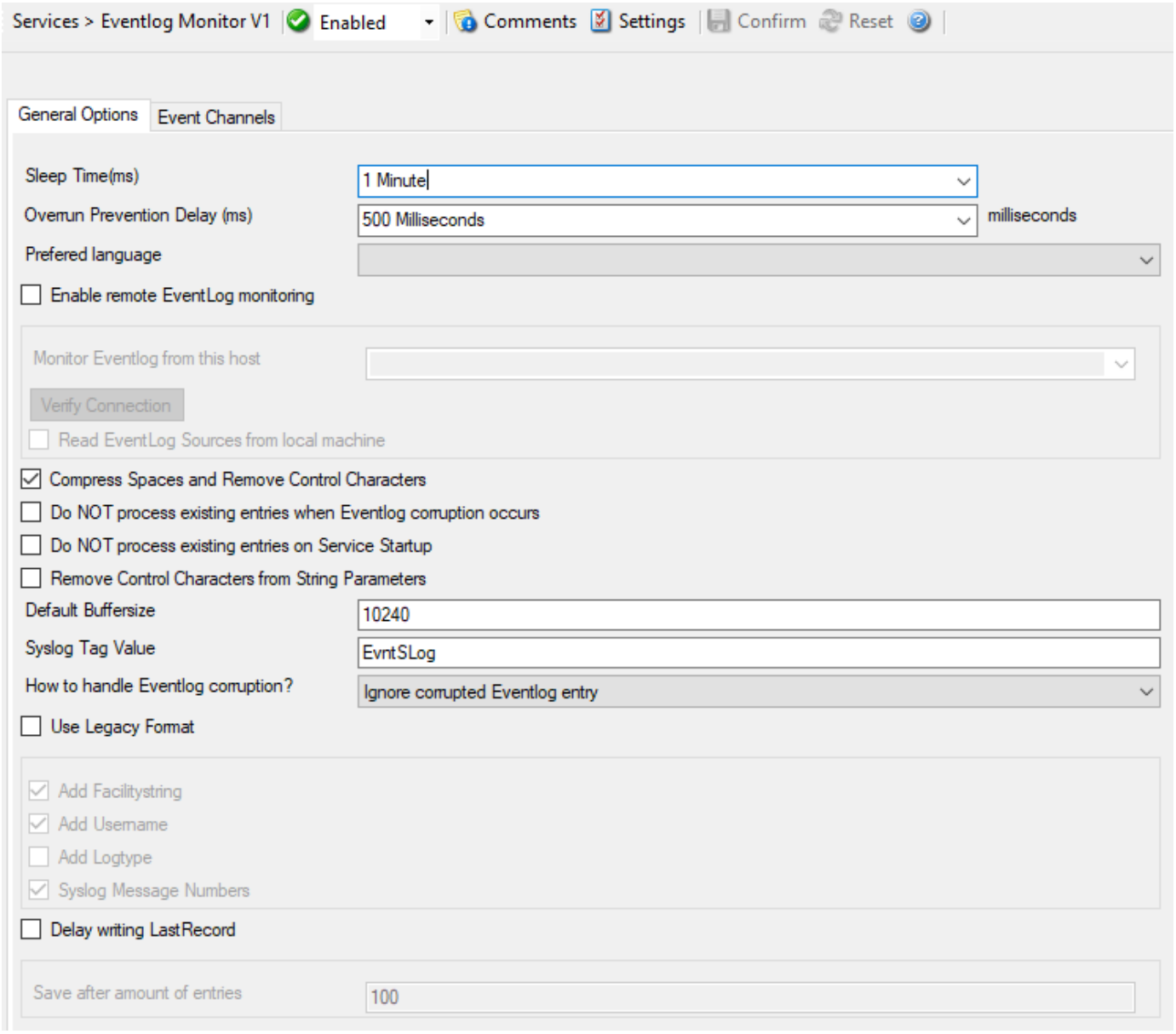
Newer Windows versions come with a considerably changed event logging system. In theory, the Event Log Monitor works with them, too. However, we know of some incompatibilities. For best results, we recommend using the event log monitor v2 service, which was specifically written for Windows Vista and newer. The Event Log Monitor described here is applicable for legacy Windows systems, and XP (where the new event logging system is not available). The Client will automatically detect and load available Event Log types during the first startup of the Event Log Monitor.

Event Log Monitor V1

- Windows XP
- Windows 2003

Event Log Monitor V2

- all modern Windows versions (Windows 10, 11, Server 2016, 2019, 2022, and newer).



- Service - Event Log Monitor V1*

General Options Tab

Sleep Time(ms)

File Configuration field:

nSleepTime

Description:

The Event Log Monitor periodically checks for new event log entries. The “Sleep Time” parameter specifies how often this happens. This value is in milliseconds.

We suggest a value of 60,000 milliseconds for the “Sleep Time”. With that setting, the Event Log Monitor checks for new events every 60 seconds. Larger periods can be specified for occasionally connected systems or if email delivery with few emails per day is intended.

Very security-aware environments might use a shorter interval. The event log monitor service is specifically designed to limit the burden on the monitored system. As such, resource usage is typically low, even with frequently run event log checks. However, we recommend not running the Event Log Monitor more often than once a second.

Overrun Prevention Delay (ms)

File Configuration field:

nPreventOverrunDelay

Description:

This property allows configuring a delay after generating an event. The time is the delay in **`milliseconds`**.

If run at a value of zero, the Event Log Monitor service generates events as fast as the machine permits. We have seen scenarios where routers and receivers are not able to keep up with this rate, resulting in packet loss. In addition, the CPU of the reporting machine is run at 100% - which is not a problem because the service runs at a low priority. However, with even a 1-millisecond delay, there is no noticeable CPU activity even when large bursts of events are forwarded. At one millisecond, the service can still generate 1000 events per second.

The default setting is an overrun protection of five millisecond, which allows roughly 200 events per second. This should be sufficient for even very busy servers.

Preferred language

File Configuration field:

nLanguageLCID

Description:

You can select a preferred language and the Eventlog Monitor will send the message in this language. It will only work if these languages are installed and message libs are available with the preferred language. If this fails, it will automatically fall back to the system default language.

Enable remote Event Log monitoring

File Configuration field:

nEnabledRemote

Description:

If enabled, the Event Log Monitor will read and process the EventLog from a remote machine. Use the verify button to make sure that the network connection is working correctly.

Please make sure that the machine, which you are going to monitor, does have File and Print Services enabled, and is accessible by this machine.

This is important as the Event Log Service will read the message libraries on the remote machine by using the default administrative shares. For this reason, the Service must be configured to run with a user who has administrative privileges/permissions on the local and remote machine. If File and Print services remain disabled, the local message libraries will used automatically instead. Note that you may experience a lot of missing message libraries in this case.

Additionally you have an option to read the Event Log Sources from the local machine. If enabled, the local message libraries will be used instead of the remote machines ones. Sometimes local Event Sources are more reliable or are required for third-party EventLog implementations.

Compress Spaces and Remove Control Characters

File Configuration field:

nCompressControlChars

Description:

This option allows you to control the control character removal and space compression. If checked, control characters (e.g. CR, LF, TAB - non printable characters in general) are removed. Also multiple spaces are compressed to a single one. By default this is checked. We recommend keeping it checked for most cases as it provides better display.

Please note that it should be unchecked if events should be forwarded via email. And it MUST be turned off if double-byte character sets are being processed (e.g. Japanese).

Do NOT process existing entries when Event Log corruption occurs

File Configuration field:

nDoNotProcessLastRecord

Description:

When this option is checked, it prevents reprocessing of the whole Windows Event Log when the log becomes corrupted or truncated. This helps avoid re-reading all existing entries after such an event.

Do NOT process existing entries on Service Startup

File Configuration field:

szSyslogTagValue

Description:

When this option is checked, it prevents from reprocessing of the whole Windows Event Log when the EventReporter / MonitorWare Agent service is restarted.

Remove Control Characters from String Parameters

File Configuration field:

nRemoveControlCharsFromParams

Description:

Enable this option to remove control characters like carriage return or line feeds from parameter strings and category names in Windows Events.

Default Buffersize

File Configuration field:

nDefaultBuffer

Description:

The default Buffersize is 10k. This value will be increased or decreased dynamically if necessary. If you want to use third-party applications like NetApp you must increase the Buffersize manually (minimum 65k), because dynamic adjusting is not possible with them.

SyslogTag Value

File Configuration field:

szSyslogTagValue

Description:

The SyslogTag Value determines the SyslogTag that is used when forwarding Events via syslog. This is useful, if you want to determine later, what kind of syslog message this is, perhaps because you log Event Logs and syslog into the same database.

How to handle Eventlog corruption

File Configuration field:

nEventLogCor

- 0 = Retry processing from beginning
- 1 = Ignore corrupted Eventlog entry
- 2 = Clear all events from Eventlog

Description:

Sometimes it can occur that Event Log messages are corrupted and cannot be read correctly. This usually happens if someone tampered with the Event Log or if you are processing the Eventlog for the first time. In cases like this, you can automatically handle the situation with this option. You have the following options:

- Retry processing Event Log from the beginning: in this case the complete Eventlog will be processed again.
- Ignore corrupted Event Log entry (default): the affected Eventlog entry will be ignored and processing will continue.
- Clear all Events from the Event Log: the Event Log will be cleared completely and new Events hopefully don't get corrupted before they are processed.

Use Legacy Format

File Configuration field:

bUseLegacyFormat

Description:

This option enhances compatibility to scripts and products working with previous versions of EventReporter. The legacy format contains all Windows Event Log properties within the message itself.

The new format includes the plain text message only. The additional information fields (like event ID or event source) are part of the XML formatted event data. If the new format is used, we highly recommend sending or storing information in XML format. This is an option in each of the action properties (of those actions that support it – the write to database option for example always stores the fields separated, so there is no specific option to do so).

Legacy format is meant to support existing parser scripts. We encourage you to use the new, XML-bound format for new implementations. Legacy format will be maintained in future releases to support backward compatibility, but it is no longer actively being developed. There are some shortcomings in legacy format, which can lead to issues when building or operating a log parser. These shortcomings are by design. We will not change this in legacy format - the solution is to use the new format. After all, the new format was created in order to address the issues with legacy format.

Add Facility String

File Configuration field:

bAddFacilityString

Description:

If checked, facility identification is prepended to the message text generated. This parameter enhances compatibility with existing Syslog programs and greatly facilitates parsing the generated entries on the Syslog server. We strongly encourage users to use this enhancement.

This setting only applies if the "Use Legacy Format" option is checked. Otherwise, it does not have any meaning and consequently cannot be configured in that case.

Add Username

File Configuration field:

nAddUserName

Description:

If checked, the Windows user that generated the event log entry is transmitted. If unchecked, this information is not forwarded. This is a configurable option for customers who have written parsing scripts for a previous format which did not contain Usernames. This option must also be unchecked if MoniLog is used.

This setting only applies if the “Use Legacy Format” option is checked. Otherwise, it does not have any meaning and consequently cannot be configured in that case.

Add Logtype

File Configuration field:

nAddLogType

Description:

If checked, then log types e.g. system, security etc. is prepended to the generated message.

This setting only applies if the “Use Legacy Format” option is checked. Otherwise, it does not have any meaning and consequently cannot be configured in that case.

Syslog Message Numbers

File Configuration field:

bShowSyslogMsgNbr

Description:

If checked, a continuously advancing message number is prepended to the generated message. This is useful for Syslog delivery to make sure that all messages have been received at the remote server

This setting only applies if the “Use Legacy Format” option is checked. Otherwise, it does not have any meaning and consequently cannot be configured in that case.

Delay writing LastRecord

File Configuration field:

nEnableLastRecordDelay

Description:

Enables the LastRecord writeback delay to the configured properties below.

Save after amount of entries

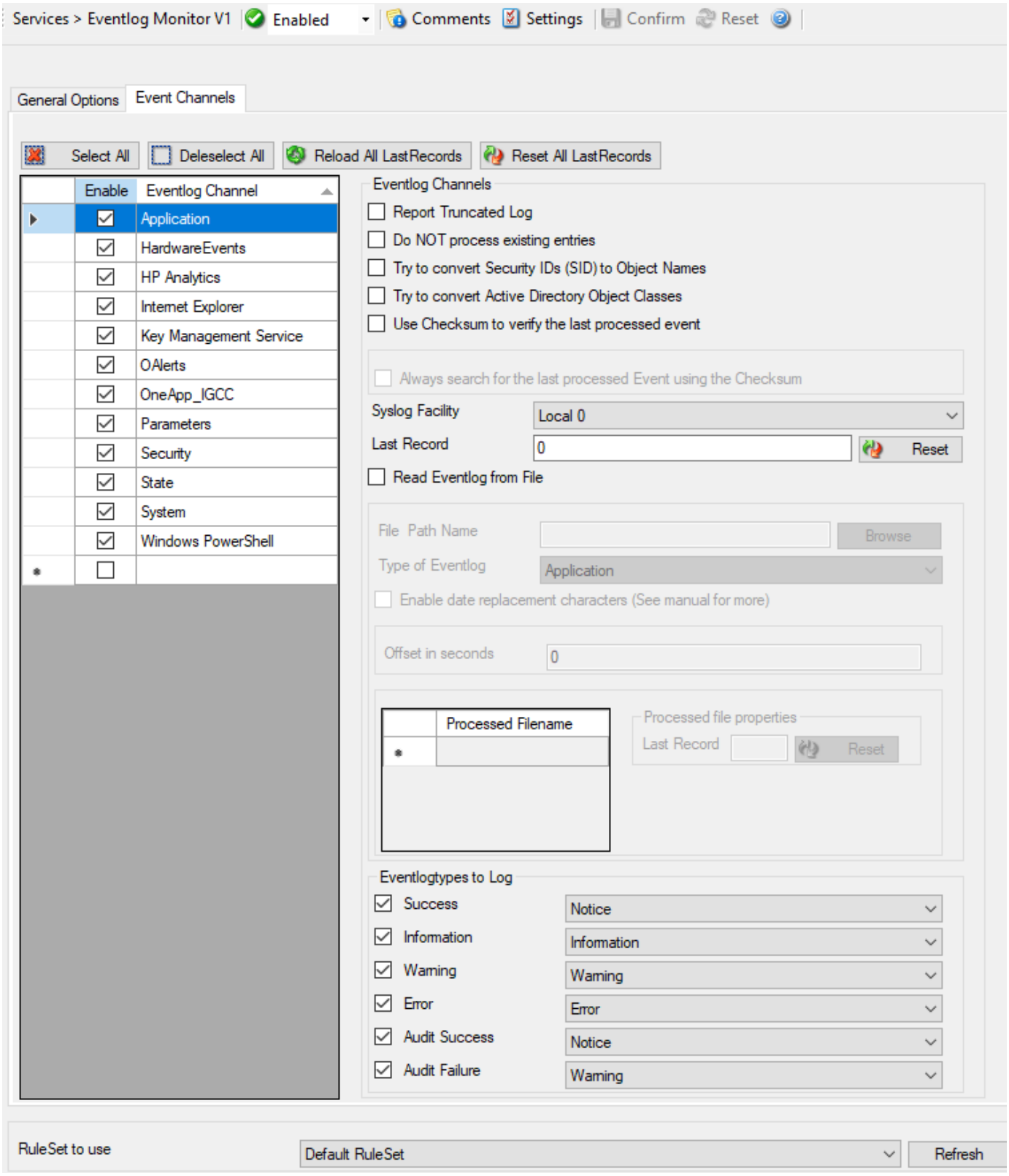
File Configuration field:

nLastRecordDelayCount

Description:

The LastRecord will be written after the amount of processed event log entries that are specified here.

Event Channels Tab



- Service - Event Log Monitor V1 Channels Tab*

Event Log Channels

They are basically a list of the different log types. The corresponding log is only be processed if the respective “Enable” checkbox is checked. The parameters are common to all logs and are explained only once.

Report Truncated Log

File Configuration field:

bReportTruncatedLog

Description:

Windows event logs can be truncated programmatically or via the Windows Event Viewer program. When a log is truncated, all information is erased from it. Any entries not already processed by the service are lost.

The service detects event log truncation. If “Report Truncated Log” is checked, it generates a separate message stating the truncation. This option is most useful in environments where truncation is not expected and as such might be an indication of system compromise.

If you regularly truncate the Windows Event Logs as part of your day-to-day operation, we suggest you turn this option off. In this case, we also recommend using a short sleep period (for example 10,000 which is 10 seconds) to avoid losing log entries.

Do NOT process existing entries

File Configuration field:

nNoExistingEntries

Description:

If you do not want to get a dump of an existing specific Windows Event Log then use this option. When MonitorWare Agent / EventReporter are restarted it will start processing after that last entry and do not look for the previous entries.

Try to convert Security IDs (SID) to Object Names

File Configuration field:

nTryConvSIDtoObj

Description:

With this option you can convert Security ID’s (SIDs) to object names. You can enable this feature in the advanced configuration of each event log type in the Event Log Monitor service. Simple check the “Try to convert Security IDs (SID) to Object Names” option.

Note that only the Security event log has this feature enabled by default. For all other event log types this feature is disabled by default.

Try to convert Active Directory Object Classes

File Configuration field:

nTryConvertDsClasses

Description:

With this option you can convert ActiveDirectory Schema GUID’s from Security Events on Domain Controllers to object names. For example Event 565, which usually has a lot of these Schema GUID’s! The GUID’s are internally cached to speed up EventLog processing operations.

Note that only the Security event log has this feature enabled by default. For all other event log types this feature is disabled by default.

Use Checksum to verify the last processed event

File Configuration field:

nEventUseChecksum

Description:

A checksum of the last processed Event will be stored along with the LastRecord of an event log. This checksum is checked during each iteration. If the checksum does not match , we consider the EventLog has

been altered, cleared, or something else happened. In this case the EventLog is being reprocessed from the beginning.

Please note: This option will prevent you from modifying the LastRecord value. If you do, the whole EventLog will be reprocessed! Please note that this behavior is by design and cannot be avoided. So we recommend to use this feature only if you intend to double check if the LastRecord value is valid.

Always search for the last processed Event using this Checksum

File Configuration field:

nEventScanLastEventByChecksum

Description:

Usually, the last processed Event is determined by the LastRecord value. If the Checksum to verify the last processed Event is activated, then the option to always search for the last processed Event using the Checksum is available. When activated, the last processed Event will also be always determined by the Checksum, not the LastRecord value.

Syslog Facility

File Configuration field:

nFacility

Description:

The syslog facility to map information units stemming from this log to. Most useful if the message is to forward to a Syslog daemon.

Last Record

File Configuration field:

nLastRecord

Description:

Windows Event Log records are numbered serially, starting at one. The service records the last record processed. This textbox allows you to override this value.

Use it with caution!

If you would like a complete dump of a specific Windows Event Log, reset the “Last Record” to zero with the reset button. If you missed some events, simply reset it to some lower value than currently set. It is possible to set “Last Record” to a higher value. This suspends event reporting until that record has been created. We strongly discourage to use this feature unless definitely needed.

Read Eventlog from File

File Configuration field:

nReadFromFile

Description:

When enabled, the Eventlog is read from a file instead from the system.

File Path Name

File Configuration field:

szLogFileName

Description:

It defines that which file to be read, only available when “Read Eventlog From File” is enabled.

Type of Eventlog

File Configuration field:

szLogType

Description:

It defines as which type of event log from file is handled. This is important to read the correct message libs from the system.

Enable date replacement characters

File Configuration field:

nEnableDateReplacements

description:

Allow the use of dynamic files/paths when using evt files. The same replacement characters as in the FileMonitor apply to this feature. A configured filename may look like this: `C:\temp\evt_%Y%m%d.evt` and would be replaced with `C:\temp\evt_20130101.evt`.

To support changing log file names, there are replacement characters available within the file name. These are:

- %y Year with two digits (e.g. 2002 becomes “02”)
- %Y Year with 4 digits
- %m Month with two digits (e.g. March becomes “03”)
- %M Minute with two digits
- %d Day of month with two digits (e.g. March, 1st becomes “01”)
- %h Hour as two digits
- %S Seconds as two digits. It is hardly believed that this ever be used in reality.
- %w Weekday as one digit. 0 means Sunday, 1 Monday and so on.
- %W Weekday as three-character string. Possible values are “Sun”, “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”. This replacement character is most useful for DHCP log files.
- %generatedfilename% It contains the fully generated filename (Can be useful for filtering).
- %msgsep% Only available if enable in the advanced option of the File Monitor. This value contains the current used message separator. This is useful if you want to reconstruct messages where the separator is part of the message.
- %msgseplast% Only available if enable in the advanced option of the File Monitor. This value contains the last used message separator. This is useful if you want to reconstruct messages where the separator is part of the message.

Offset in seconds

File Configuration field:

nEnableDateReplacementsOffset

Description:

When “Enable date replacement characters” is enabled, the current date will be used to generate the filenames. However in certain cases, there is a need to generate filenames with past or future dates. Negative values will generate past filenames, while positive values will generate filenames in the future. For example if you want to generate filenames from yesterday (24 hours back), use -84600 as offset.

Event Types to Log

These checkboxes allow local filtering of the event log. Filtering is based on the Windows event type. There is a checkbox corresponding to each Windows event type. Only checked event types will be processed. Unchecked ones will be ignored.

Filtering out unnecessary log types at this level enhances system performance because no information units will be generated and passed to the rule engine. Thus, Adiscon strongly recommends dropping unnecessary log types.

Ruleset to use

File Configuration field:

szRuleSetName

Description:

Name of the ruleset to be used for this service. The RuleSet name must be a valid RuleSet.

Note

If you intend to make the Event ID part of the actual Syslog message while forwarding to a Syslog server, adjust the Event Log Monitor output format or the Forward Syslog action message format accordingly.

The Event Log Monitor caches messages libraries. This greatly speeds up processing, but causes memory consumption for the cached libraries. By default, libraries are cached for 30 minutes. If memory consumption is too high, you may consider to lower the cache period. The cache is global to all event log monitors. As such, its size must be changed in the Engine specific Options Tab. Here you find the Resource Library Cache Options

Event Log Monitor V2

This dialog configures the Windows Event Log Monitor V2 service.

Eventlog Monitor V2

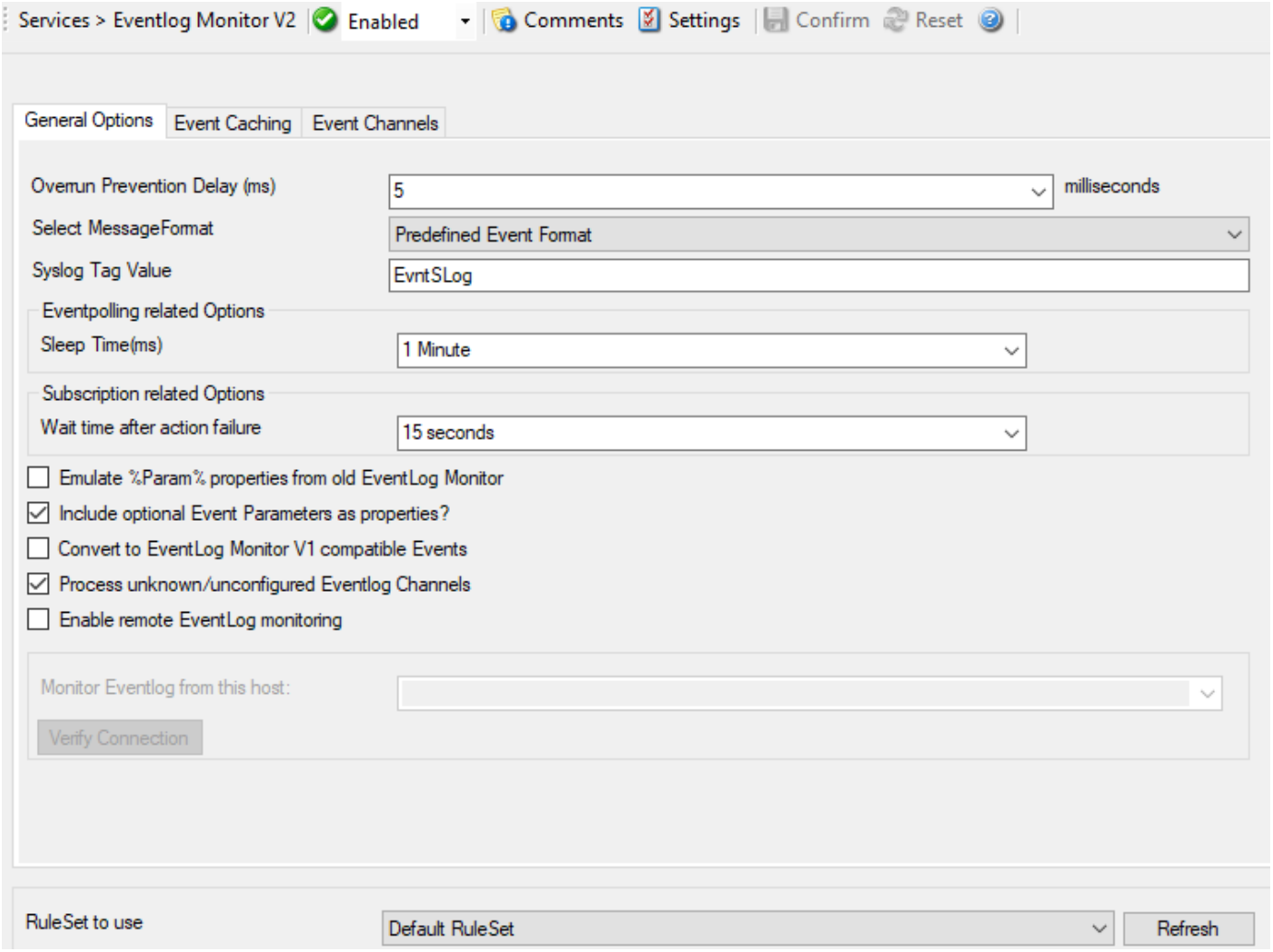
- all modern Windows versions (Windows 10, 11, Server 2016, 2019, 2022, and newer).

Eventlog Monitor

- Windows XP
- Windows 2003

The V2 Eventlog Monitor provides the ability to monitor so-called “log channels”. Each channel can work either in polling or subscription mode. In subscription mode, we are automatically notified by the operating system whenever a new event is logged. In traditional polling mode, we periodically check the channel. In both cases, it is possible for a user to re-set the channel reporting to an older event (parameter “Last Record”).

Both of these functionalities are implemented by periodically iterating over the configured channels. The frequency is controlled by the “Sleep Time” parameter.



- Service - Event Log Monitor V2 - General Options*

General Options Tab

Overrun Prevention Delay (ms)

File Configuration field:

nPreventOverrunDelay

Description:

This property allows configuring a delay after generating an event. The time is the delay in **`milliseconds`**.

If run at a value of zero, the Event Log Monitor service generates events as fast as the machine permits. We have seen scenarios where routers and receivers are not able to keep up with this rate, resulting in packet loss. In addition, the CPU of the reporting machine is run at 100% - which is not a problem because the service runs at a low priority. However, with even a 1-millisecond delay, there is no noticeable CPU activity even when large bursts of events are forwarded. At one millisecond, the service can still generate 1000 events per second.

The default setting is an overrun protection of five millisecond, which allows roughly 200 events per second. This should be sufficient for even very busy servers.

Select Message Format

File Configuration field:

nFormatType

- 0 = XML Format
- 1 = Predefined EventFormat

Description:

With this option you can choose whether the Events will be extracted in “Raw XML Format” or in the “Predefined Event Format”.

The XML format is the exact representation of the XML Stream returned by the EventLog System. **Please note that it only contains EventLog data and not a formatted message.**

The “Predefined Event Format” is what the Event in the event viewer looks like.

Copy Format into Property

If enabled, a second message format can be stored into a custom property.

Select Message Format

File Configuration field:

nCopyFormatIntoProperty

Description

Select which message format should be stored into the custom property.

Store into Property

File Configuration field:

szCopyFormatIntoProperty

Description

The custom message property, for the “Copy Format into Property” Option.

SyslogTag Value

File Configuration field:

szSyslogTagValue

Description:

The SyslogTag Value determines the SyslogTag that is used when forwarding Events via syslog. This is useful, if you want to determine later, what kind of syslog message this is, perhaps because you log EventLogs and syslog into the same database.

Eventpolling related Option: Sleep Time(ms)

File Configuration field:

nSleepTime

Description:

As said in the overview, this controls iteration over the configured channels. The value is specified in milliseconds.

For channels configured to use Polling Mode, the “Sleep Time” parameter specifies how often they are processed. Note that when multiple channels are set to polling mode, they are processed one after another. So there is a somewhat larger delay in processing than given by the “Sleep Time” parameter. The total frequency depends on how busy all polling channels are.

For channels configured to subscription mode, the “Sleep Time” interval will only influence how often a potential reset of “Last Record” is checked. Other than that, it has no effect on the event delivery rate.

We suggest a value of 60,000 milliseconds for the “Sleep Time”. With that setting, the Event Log Monitor checks for new events every 60 seconds. Larger periods can be specified for occasionally connected systems or if email delivery with few emails per day is intended.

Very security-aware environments might use a shorter interval. The event log monitor service is specifically designed to limit the burden on the monitored system. As such, resource usage is typically low, even with frequently run event log checks. However, we do NOT recommend running the Event Log Monitor more often than once a second.

Note that it is almost always an error to use a “Sleep Time” value of 0. The main processing loop of the EventLog Monitor V2 would re-run without any delay and would cause a very high CPU usage, close to 100%. For these reasons, newer versions of the product will no longer permit to use a “Sleep Time” of zero and automatically change it to one

Subscription related Option: Wait time after action failure

File Configuration field:

nSubscriptionSleepTime

Description:

Adds some extra wait time (Delay) if an action failed to process. Without the delay, the subscription would immediately process the last event again. In some cases a reasonable delay before a retry is needed.

Emulate %Param% properties from old EventLog Monitor

File Configuration field:

nEmulateParameters

Description:

This option emulates the %Param% properties, which were often used in the old EventLog Monitor. The new EventLog implementation does not support them in the same way anymore. The Event Log Monitor V2 is still able to provide parameters in the “old style” format, what means that log analysis scripts can receive a consistent stream of data for both new style and old style Windows events.

Include optional Event Parameters as properties?

File Configuration field:

nIncludeEventParameters

Description:

If enabled, the < EventData> node from the raw XML stream (Eventlog entry) will be searched for variables. If variables with names are found, they will be set as Properties with their variable name automatically. If the variable does not have a name, it will be set to a common name like “Param1, Param2 …. ParamX”.

Convert to EventLog Monitor V1

File Configuration field:

nConvertToEventLogMonitorV1

Description:

This option maps the EventID's from the Security EventLog back to V1 (Windows 2000/2003). The internal InforUnitID is also changed to V1. This option helps postprocessing EventLog V1 and V2 events equally.

Process unknown/unconfigured Eventlog Channels

File Configuration field:

nEnableUnknownChannels

Description:

If enabled, unconfigured Eventlog Channels (Those found on the system, but not listed in the Eventlog Channels list) will automatically be processed.

Enable remote EventLog monitoring

File Configuration field:

nEnabledRemote

Description:

If enabled, EventLog Monitor will connect to a remote machine to process the EventLog. Please note that the RPC Service needs to be installed on the remote machine, and the Service has to be configured with a network user that has sufficient access rights.

Monitor Eventlog from this host

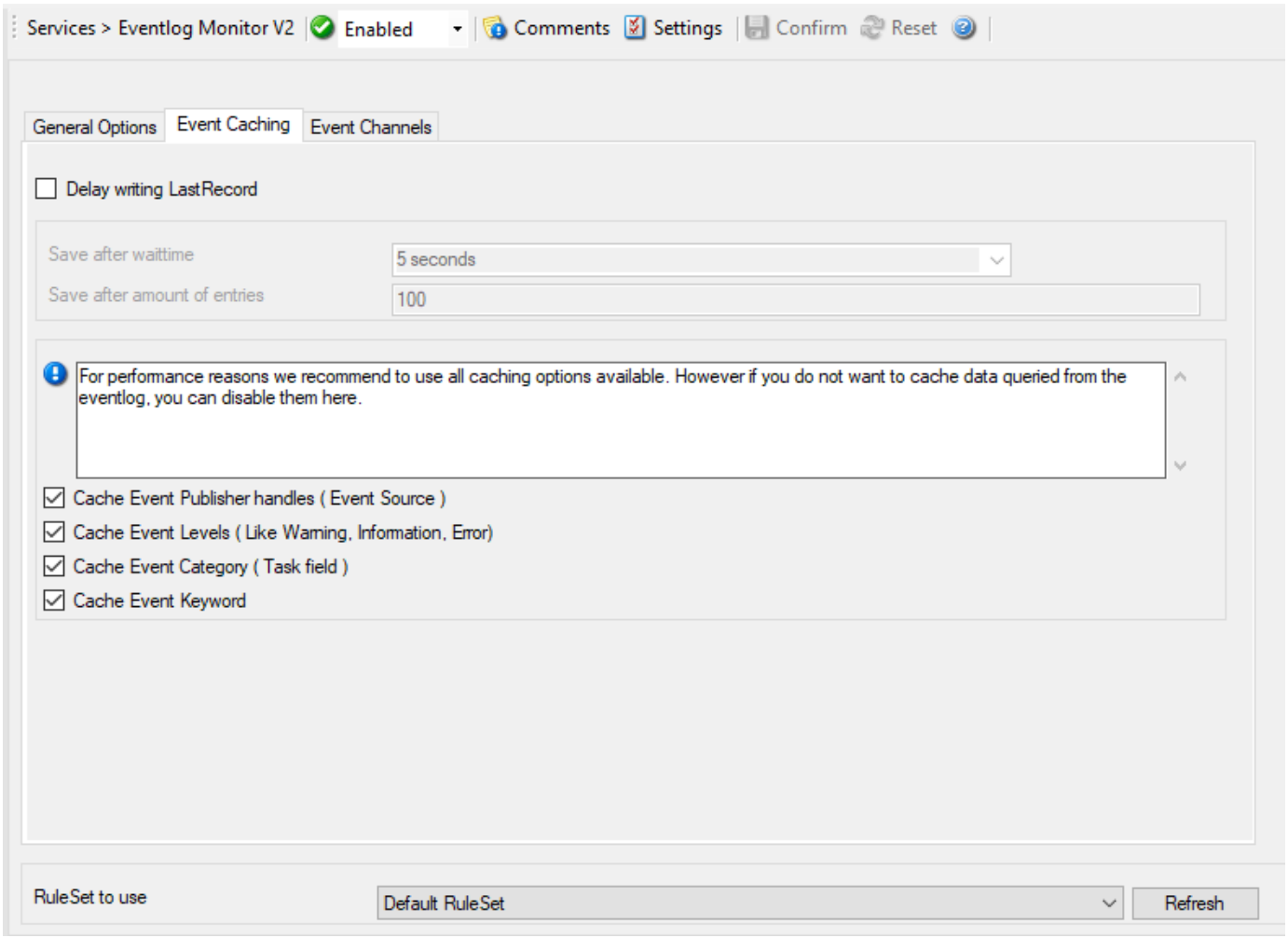
File Configuration field:

szServerName

Description:

The hostname of the remote server to be monitored by EventLog Monitor.

Event Caching Tab



- Service - Event Log Monitor V2 - Event Caching*

Delay writing LastRecord

File Configuration field:

nEnableLastRecordDelay

Description:

Enables the LastRecord writeback delay to the configured properties below.

Save after waittime

File Configuration field:

nLastRecordDelayTime

Description:

Regardless of the amount of processed event log entries, the lastrecord value will delayed for this waittime period.

Save after amount of entries

File Configuration field:

nLastRecordDelayCount

Description:

Regardless of the configured waittime period, the LastRecord will be written after the amount of processed event log entries that are specified here.

Cache Event Publisher handles (Event Source)

File Configuration field:

nCacheEventPublisher

Description:

All publisher sources will be kept open once loaded until the application / service is stopped. This increases processing speed for events from same sources.

Cache Event Levels (Like Warning, Information, Error)

File Configuration field:

nCacheEventLevel

Description:

If enabled the textual representations for event levels will be cached.

Cache Event Category (Task field)

File Configuration field:

nCacheEventCategory

Description:

If enabled, all textual representations for event categories will be cached.

Cache Event Keyword

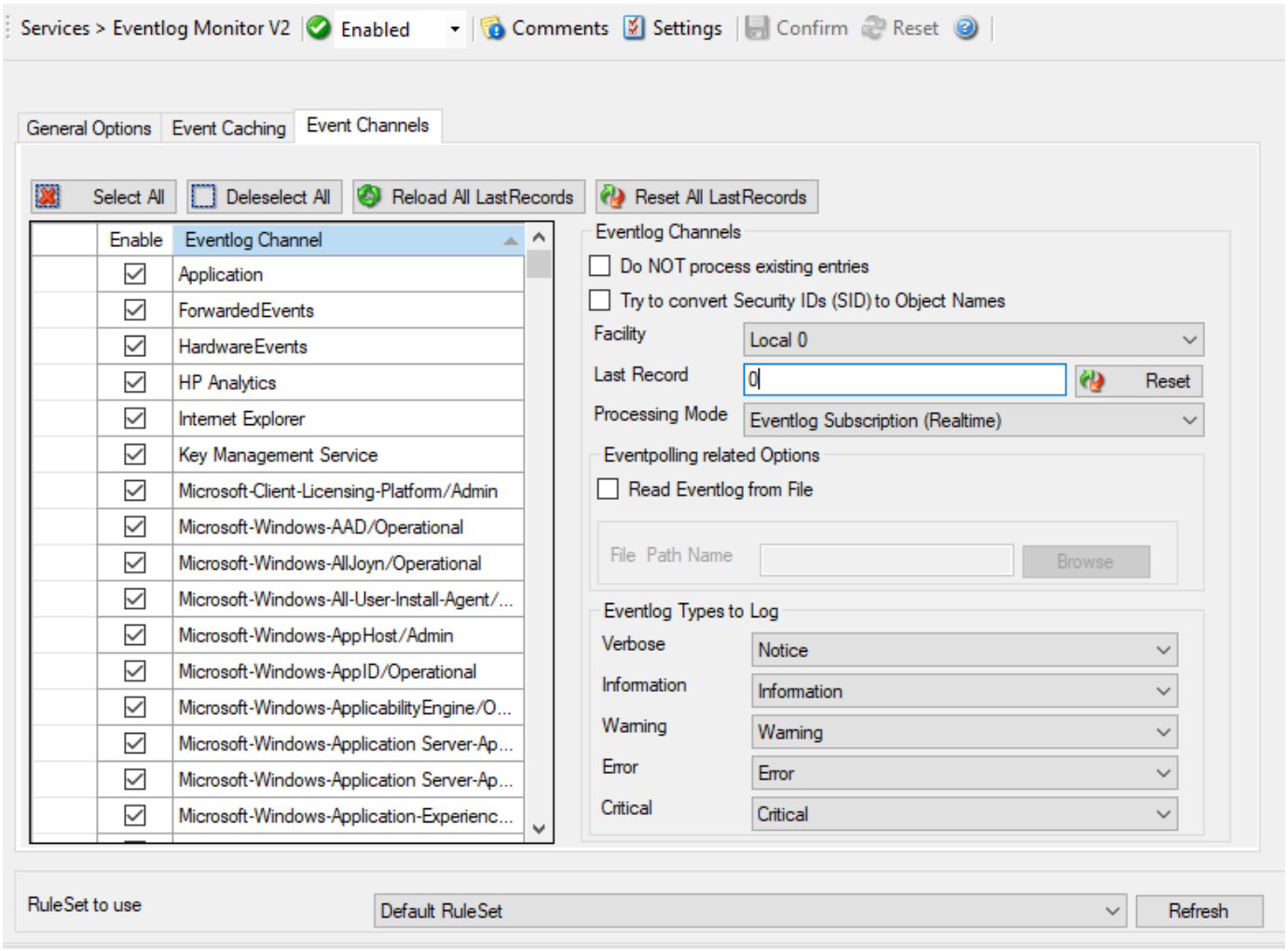
File Configuration field:

nCacheEventKeyword

Description:

If enabled, all textual representations for event keywords will be cached.

Event Channels Tab



- Service - Event Log Monitor V2 Event Channels*

The most important part of this dialog is the treeview of available Channels. It specifies which event logs are to be monitored. The monitor is set to all Channels that are currently available. There happen to be custom Channels, too, due to Applications creating them on their own. They will be added to the treeview automatically every time you re-open this configuration window.

Here you can adjust the syslog facility and the event log types. You are also able to overwrite all existing custom advanced channel configurations with your new ones.

Channels which are checked in the table will be processed. Channels which are unchecked are kept in the configuration, but are not processed.

Do NOT process existing entries

File Configuration field:

nNoExistingEntries

Description:

If you do not want to get a dump of an existing specific Windows Event Log then use this option. When MonitorWare Agent / EventReporter are restarted it will start processing after that last entry and do not look for the previous entries.

Try to convert Security IDs (SID) to Object Names

File Configuration field:

nTryConvSIDtoObj

Description:

With this option you can convert Security ID's (SIDs) to object names. You can enable this feature in the advanced configuration of each event log type in the Event Log Monitor service. Simple check the "Try to convert Security IDs (SID) to Object Names" option.

Note that only the Security event log has this feature enabled by default. For all other event log types this feature is disabled by default.

Facility

File Configuration field:

nFacility

Description:

The syslog facility to map information units stemming from this log to. Most useful if the message is to forward to a Syslog daemon.

Last Record

File Configuration field:

szXMLBookmark

Description:

Windows Event Log records are numbered serially, starting at one. The service records the last record processed. This textbox allows you to override this value. **Use it with caution!**

If you would like a complete dump of a specific Windows Event Log, reset the "Last Record" to zero. If you missed some events, simply reset it to some lower value than currently set. It is possible to set "Last Record" to a higher value. This suspends event reporting until that record has been created. We strongly discourage to use this feature unless definitely needed.

Processing Mode

File Configuration field:

nApiReadMode

- 0 = Subscription Readmode (Real-time)
- 1 = Polling Readmode (Sleeptime)

Description:

There are two processing modes available, first the default processing mode is "EventLog Subscription" which processes Events in real time. This means events are send to MonitorWare Agent by the OS as they happen, there is no delay at all. The other processing mode called "Eventlog Polling" and is similar to the method used in the old EventLog Monitor. The EventLog is checked and processed periodically controlled by the sleeptime. However using the polling method, you enable the "Read EventLog From File" option.

Eventpolling related Options

Read Eventlog from File

File Configuration field:

nReadFromFile

Description:

When enabled, the Eventlog is read from a file instead from the system.

File Path Name

File Configuration field:

szLogFileName

Description:

It defines which is file to be read, only available when “Read Eventlog From File” is enabled.

Event Types to Log

These checkboxes allow local filtering of the event log. Filtering is based on the Windows event type. There is a checkbox corresponding to each Windows event type. Only checked event types will be processed. Unchecked ones will be ignored.

Filtering out unnecessary log types at this level enhances system performance because no information units will be generated and passed to the rule engine. Thus, Adiscon strongly recommends dropping unnecessary log types.

RuleSet to use

File Configuration field:

szRuleSetName

Description:

Name of the ruleset to be used for this service. The RuleSet name must be a valid RuleSet.

Filter Conditions

Filter conditions specify **when** a rule should match. If the condition of a rule evaluates to true, EventReporter executes the actions configured in that rule.

Why filters matter

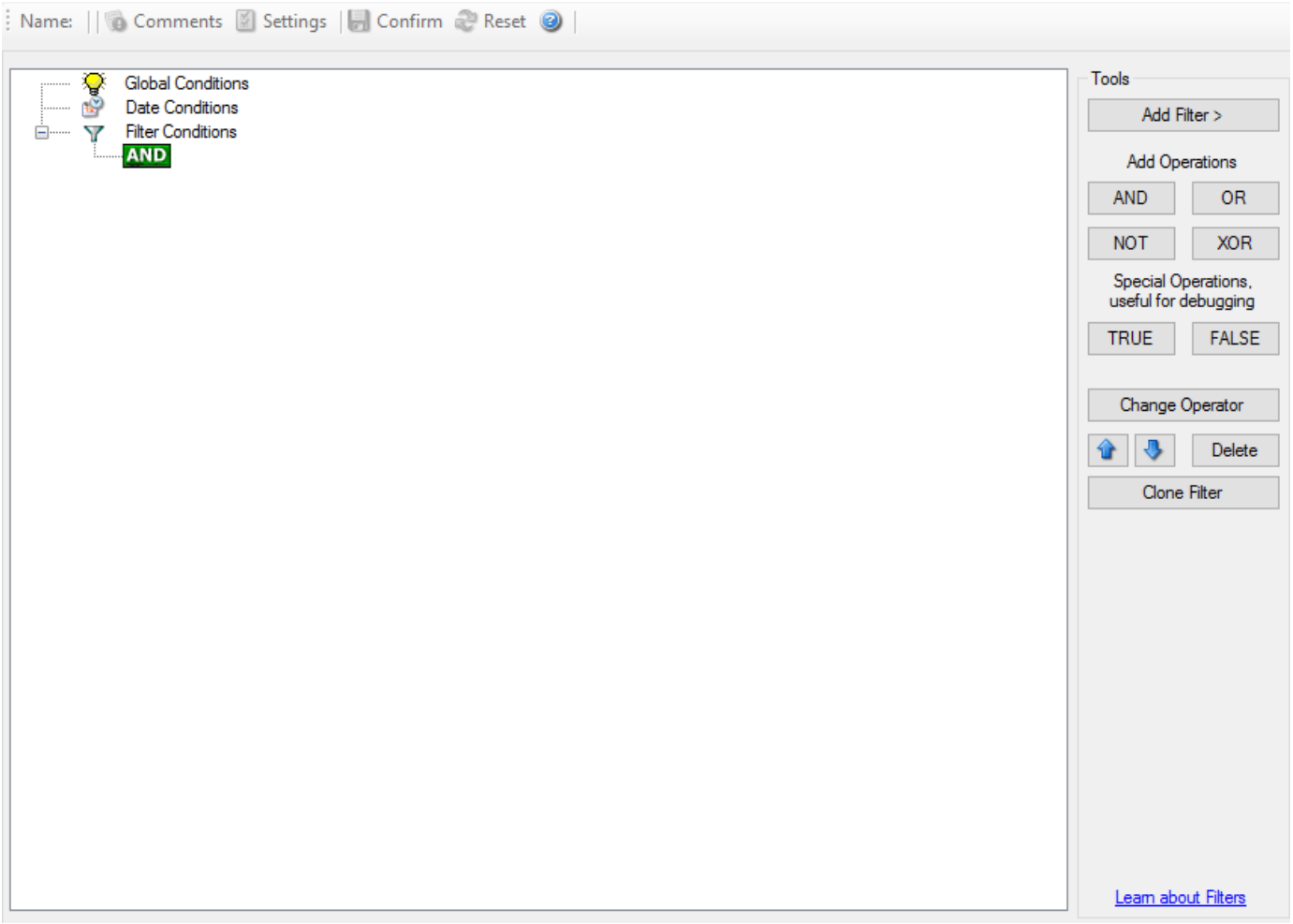
Filters let you reduce noise and react only to the events that matter. Typical criteria include:

- event log name or channel
- event source
- event ID
- severity or type
- message content
- user or computer context

Important behavior

- An empty top-level `AND` condition evaluates as true.
- That means a rule with no additional filters matches every event that reaches it.
- String matching is case-sensitive unless the specific filter documents a different behavior.

Example

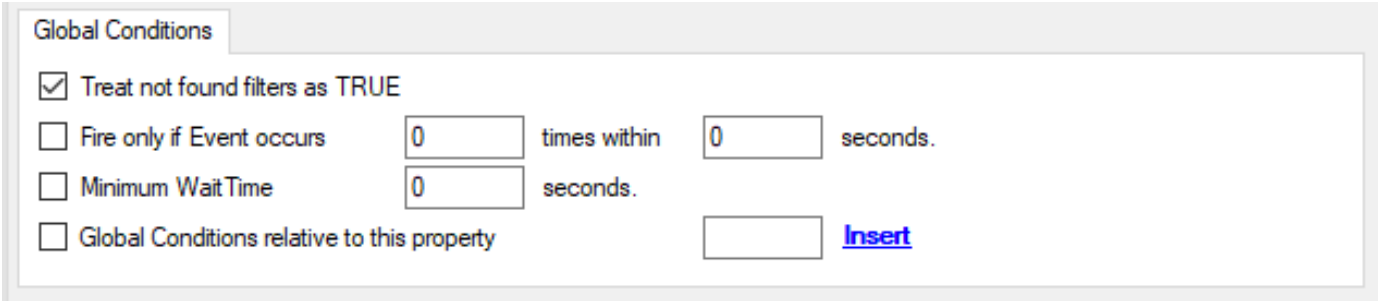


Use broad filters first, then narrow them until the rule matches exactly what you intend.

Detailed filter references

Global Conditions

Global Conditions apply to the rule as whole. They are automatically combined with a logical “AND” with the conditions in the filter tree.



- Global Conditions*

Treat not found Filters as TRUE

If a property queried in a filter condition is not present in the event, the respective condition normally returns “FALSE”. However, there might be situations where you would prefer if the rule engine would evaluate this to “TRUE” instead. With this option, you can select the intended behavior. If you check it, conditions with properties not found in the event evaluates to “TRUE.

Fire only if Event occurs

This is kind of the opposite of the “Minimum WaitTime”. Here, multiple events must come in before a rule fires. For example, this time we use a ping probe. Ping is not a very reliable protocol, so a single ping might be lost. Thus, it may not be the best idea to restart some processes just because a single ping failed. It would be much better to wait for repetitive pings to fail before doing so.

Exactly this is why the “Fire only if Event Occurs” filter condition is made for. It waits until a configured amount of the same events occurs within a period. Only if the count is reached, the filter condition matches and the rule can fire.

Note: If you used previous versions of the product, you might remember a filter called “Occurrences”. This has just been renamed.

Minimum Wait Time

This filter condition can be used to prevent rules from firing too often. For example, a rule might be created to check the status of a port probe event. The port probe probes an smtp server. If the event is fired and the rule detects it, it spawns a process that tries to restart the service. This process takes some time. Maybe the SMTP gateway need some more time to fully start up so that the port probe might fail again while the problem is already taken care of. The port probe as such generates an additional event.

Setting a minimum wait time prevents this second port probe event to fire again if it is – let’s say – within 5 minutes from the original one. In this case, the minimum wait time is not yet reached and as such, the rule does not match. If, however, the same event is generated 5 hours later (with the mail gateway failing again), the rule once again fires and corrective actions are taken.

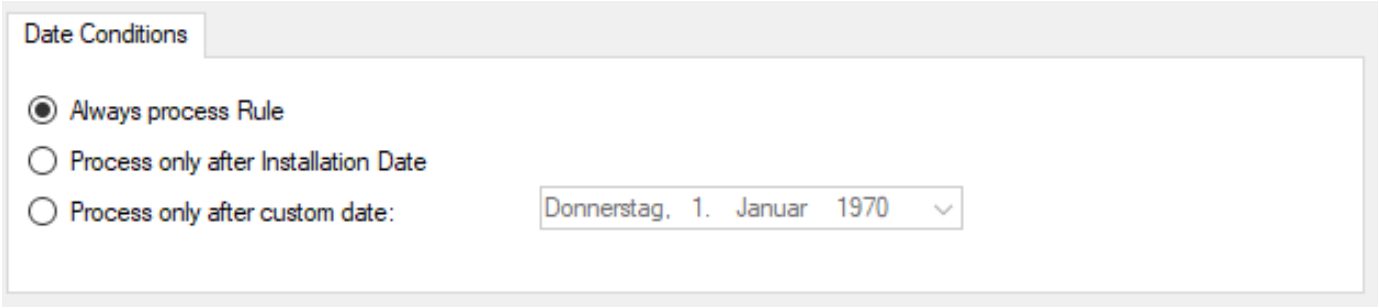
Global Conditions relative to this property

This feature enables you to control the Global Conditions based on a property.

For example take the source of a message as property. In this case, the Minimum WaitTime for example would be applied individual on each message source.

Date Conditions

Rule processing can be bound to a specific or the installation date. By default a Rule will always be processed.



- Date Conditions*

Always process Rule

No date filter will be applied

Process only after Installation Date

Rule will only be processed if message was generated after the application installation date.

Process only after custom date

Rule will only be processed if message was generated after the custom specified date.

Operators

In general, operators describes how filter conditions are linked together. The following operators can be used. Boolean operators like “AND” or “OR” can be used to create complex filter conditions.

AND:

All filters placed below must be true. Only then AND returns TRUE.

OR:

If one or both of the filters placed below is true, OR returns TRUE.

NOT:

Only one Filter can be placed below NOT operator, and if the filter evaluation is true, NOT returns FALSE.

XOR:

If one of the two filters are possible in the XOR Operator.

TRUE:

Useful for debugging, just returns TRUE.

FALSE:

Useful for debugging as well, returns FALSE.

Filters

Filters can be added under each Operation node. There are a few common filters which can be used for all services and there are special filters which only apply if a special kind of Information Unit is evaluated.

What happens with Filters that are not available in an “Information Unit”?

Every filter that is not found in an Information Unit is ignored in the filtering process. If you want to create filters specialized for types of Information Units, always make sure to add an “Information Unit Type” filter.

An example, you have one ruleset, rule and action. In the filters you have one EventID filter. Then you have two services, one Eventlog Monitor and the other is Heartbeat monitor both pointing to this ruleset. The Information Units from the Eventlog Monitor would be filtered correctly, but those from the Heartbeat monitor would not be filtered as they do not have an EventID property. The EventID filter would be ignored and the actions would be executed every time.

Note, if a filter is used that does not apply to the evaluated Info Unit, it will be just ignored. This gives you the possibility to build one filter set for several types of Information Units.

There are different types of filters, and so there a different ways in which you can compare them to a value. The following Types exist:

String:

Can be compared to another String with “=”, “Not =”, “Range Match” or through

REGEX Compare Operation

The property will be evaluated against a regular expression. Everything known in the regular expression syntax can be used to define a matching pattern.

Here are some regular expressions samples:

Regular Expression:

```
[0-9]{4,4}-[0-9]{1,2}-[0-9]{1,2} [0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}
```

Matches typical Date like `2018-11-20 12:11:01`

Regular Expression: `\n[0-9]{4,4}`

Matches Linefeed and 4-digit number.

Regular Expression: `(;|:)` Matches semicolon or a colon.

More samples and details about the Regular Expression Syntax can be found here:

https://msdn.microsoft.com/en-us/library/bb982727(v=vs.90).aspx

number:

can be compared with another number with “=”, “not =”, “<” and “>”

boolean:

can be compared to either true or false with “=” and “not =”

time:

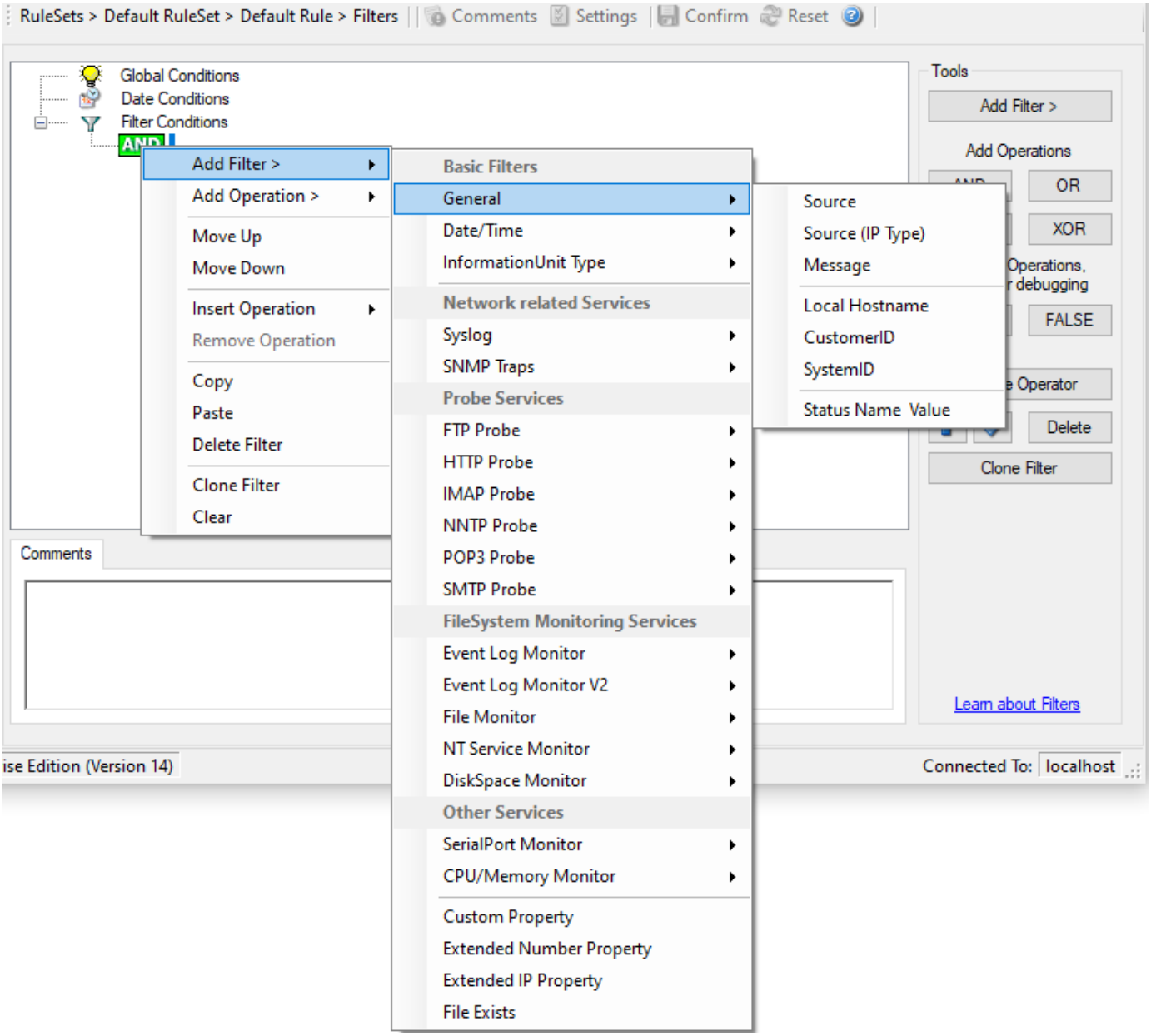
can be compared with another time but only with “=”

the list of possible filters, which can be evaluated is described in the following sections.

Basic filters

General

These are non-event log specific settings.



- Filter Conditions - General*

Source

This filter condition checks the system that generated the information unit. For example, in case of the Syslog server, this is the Syslog device sending a Syslog message.

This filter is of type string and should contain the source system name or IP address.

Source System (IP)

The IP Filter can basically work on any property, but we recommend to only use it on the %source% property, as we usually can be sure that this is contains a valid IP Address or hostname. The IP Filter can filter against hostnames and IP Addresses, hostnames are automatically resolved using the internal DNSCache (for obvious performance reasons).

This filter is of type string and should contain the source system name or IP address.

Please see the description for extended ip property for more information on how to use this property.

Message Content

The message content filter condition is very powerful. It evaluates to true if the specified content is found anywhere within the message. As there is implicit wildcarding, there is no need for extra wildcards to be specified.

The content search can be limited to a region within the message. To do so, select a starting and ending position within the string by choosing the **“contains within range”** compare operation. This can be done by specifying the start range and end range into the respective boxes.**Please note that you can enter the character position you desire in these fields. The default “Start Range” and “End Range” are set to 0.**

If you would like to search for a string just between positions 10 and 50, specify these values as start and end values, respectively. Similarly if you want to receive all logs from 192.168.0.1 then set this as:

- Property value = 192.168.0.0
- Range Start = 0
- Range End = 10

Which means 10 characters starting at zero (“192.168.0.”). Please note that the final DOT must be included. If you just used range “9”, then 192.168.010 would also match.

This filter is of type string.

CustomerID

CustomerID is of type integer provided for customer ease. For example if someone monitors his customer’s server, he can put in different CustomerIDs into each of the agents. Let us say someone monitors servers A and B. A has 5 servers all of them with CustomerID = 1 and B has 2 servers all of them with CustomerID = 2. Both A and B happen to have a server named “SERVER”. Together with the customerID, these machines are now uniquely identifiable. This is user configurable.

CustomerID (Type=Number).

SystemID

SystemID is of type integer to be used by our customer. In addition, it is user configurable.

SystemID (Type=Number).

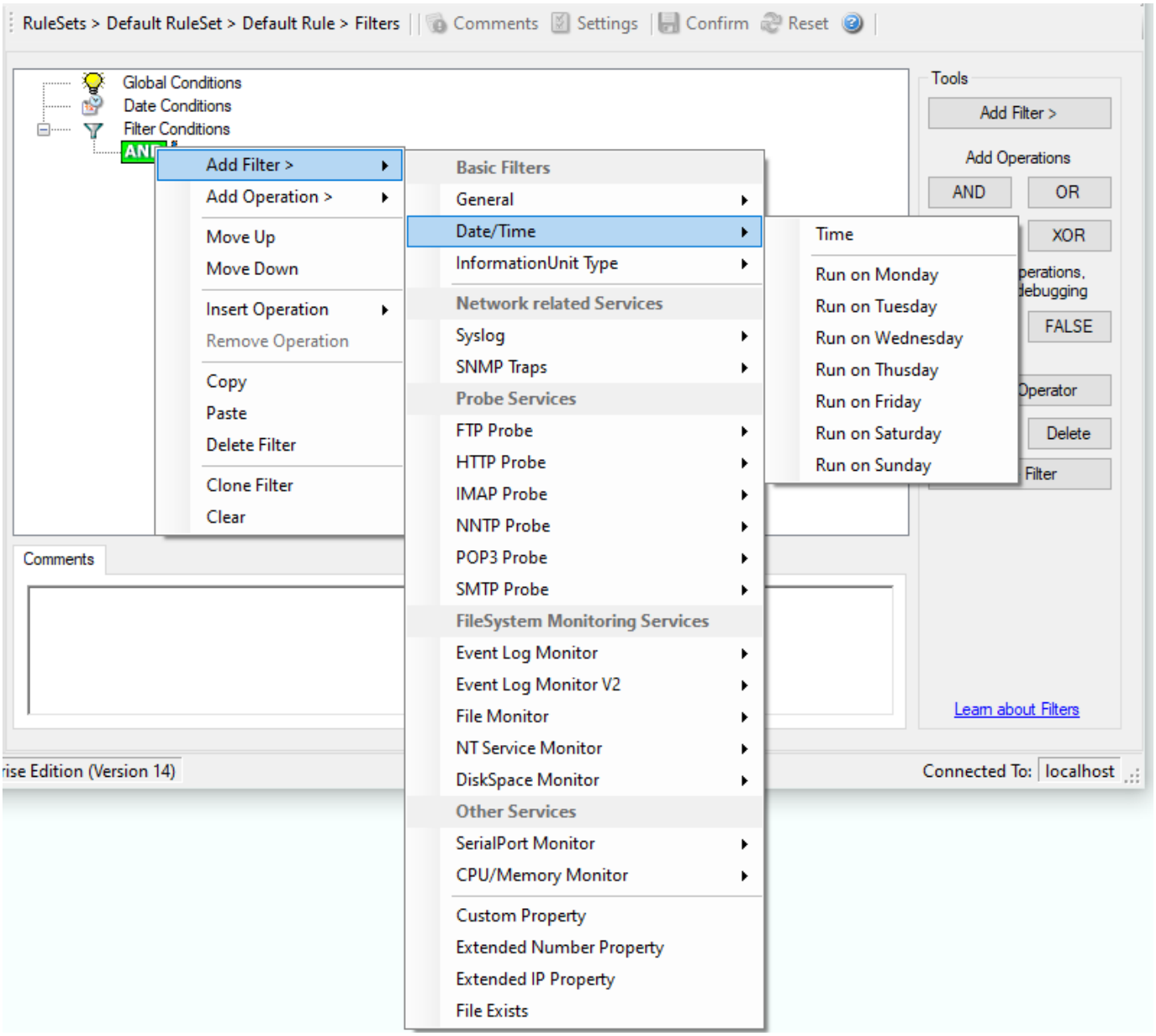
Status Name and Value

These filter type corresponds to set status action .

Status Name and Value (Type=String)

Date/Time

This filter condition is used to check the time frame and / or day of week in which an event occurred.



- Filter Conditions - Date/Time*

Time

This filter condition is used to check the period in which an event occurred. For example, a Syslog message from a Cisco router saying that it dialed up is normal if it occurs during office hours. If it occurs at night, so, it is an alerting signal and an administrator might receive notification of this event (while he might otherwise decide to discard it). This can be done with the time setting.

You can also set the timezone setting (DefaultTimemode, UTC or Localtime) for the TimeMode's (DeviceReportedTime/ReceivedTime).

Weekdays

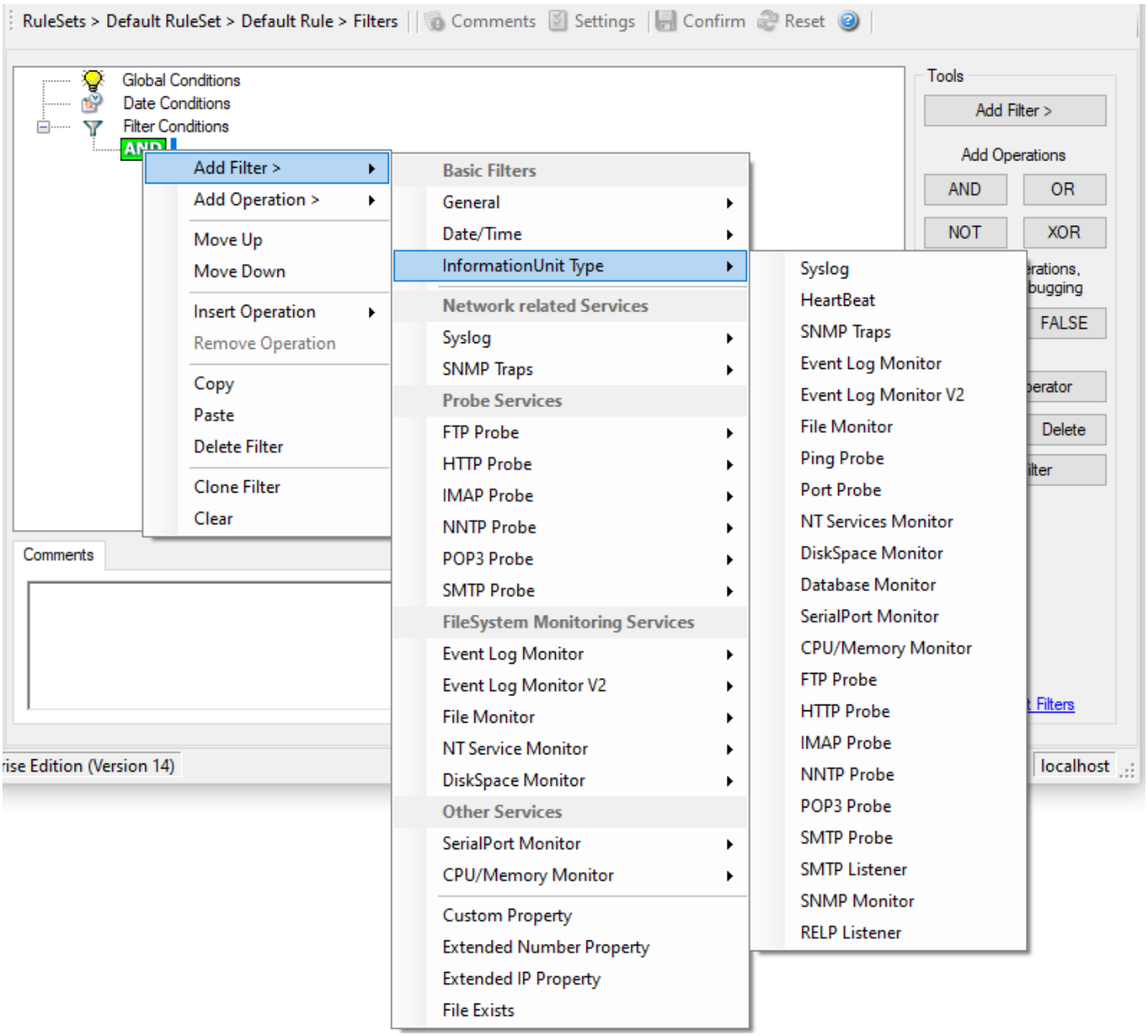
This is closely equivalent to the time filter condition, except that it is applied on a per-day basis. So it can be used to detect for example events occurring on weekends and act differently on them. The following filters are available:

1. Run on Monday (Type=Boolean)
2. Run on Tuesday (Type=Boolean)
3. Run on Wednesday (Type=Boolean)

4. Run on Thursday (Type=Boolean)
5. Run on Friday (Type=Boolean)
6. Run on Saturday (Type=Boolean)
7. Run on Sunday (Type=Boolean)

InformationUnit Type

Select the specific information if a rule should just be processed for some information unit types. This is especially useful if a specific type needs non-standard processing. There is one pre-defined filter for each possible InformationUnit Type available (shown below).



- Filter Conditions - InformationUnit Type*

The following filters are available:

1. Syslog (Type=Boolean)
2. Heartbeat (Type=Boolean)
3. SNMP Traps (Type=Boolean)
4. Event Log Monitor (Type=Boolean)
5. File Monitor (Type=Boolean)
6. Ping Probe (Type=Boolean)
7. Port Probe (Type=Boolean)
8. NT Services Monitor (Type=Boolean)
9. Disk Space Monitor (Type=Boolean)
10. Database Monitor (Type=Boolean)

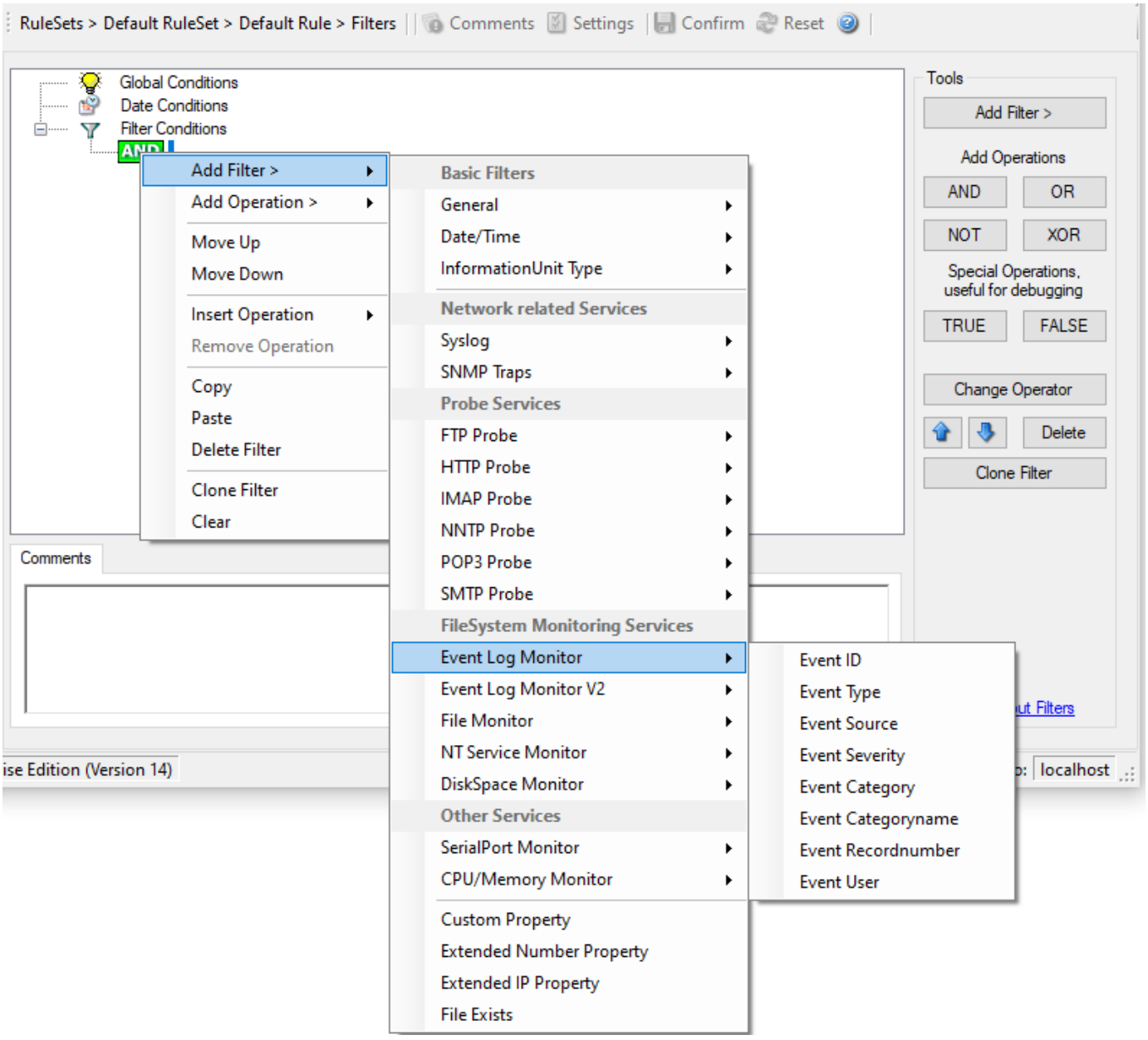
.

11 Serial Port Monitor (Type=Boolean)
.
12 CPU/Memory Monitor (Type=Boolean)
.
13 FTP Probe (Type=Boolean)
.
14 HTTP Probe (Type=Boolean)
.
15 IMAP Probe (Type=Boolean)
.
16 NNTP Probe (Type=Boolean)
.
17 POP3 Probe (Type=Boolean)
.
18 SMTP Probe (Type=Boolean)
.

Event log monitor filters

Event Log Monitor

Event Log Monitor specific filters are grouped here.



- Filter Conditions - Event Log Monitor V1*

Event ID

This is the event log ID as specified in the Windows Event Log. If enabled, the event must have the configured event ID or the rule will not match. This is an integer value.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type number.

Event Type

This is the event log type as specified in the Windows Event Log. If enabled, the event must have the configured event type or the rule will not match. The supported values can be selected from the list box.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event Source

This is the event log source as specified in the Windows Event Log. If enabled, the event must have the configured event source or the rule will not match. This is a string value. There must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event Severity

This is the event log severity as specified in the Windows Event Log. If enabled, the event must have the configured severity or the rule will not match. The supported values can be selected from the list box.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type number.

Event Category

This is the event log category as specified in the Windows Event Log. If enabled, the event must have the configured event category or the rule will not match.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type number.

Event Categoryname

This value contains the Category value as string if it can be resolved. Otherwise it contains the category number.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event Recordnumber

This value contains the internal event record number. Please note that if the event log has been truncated before, it may not start with 0 or 1 but a higher number.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type number.

Event User

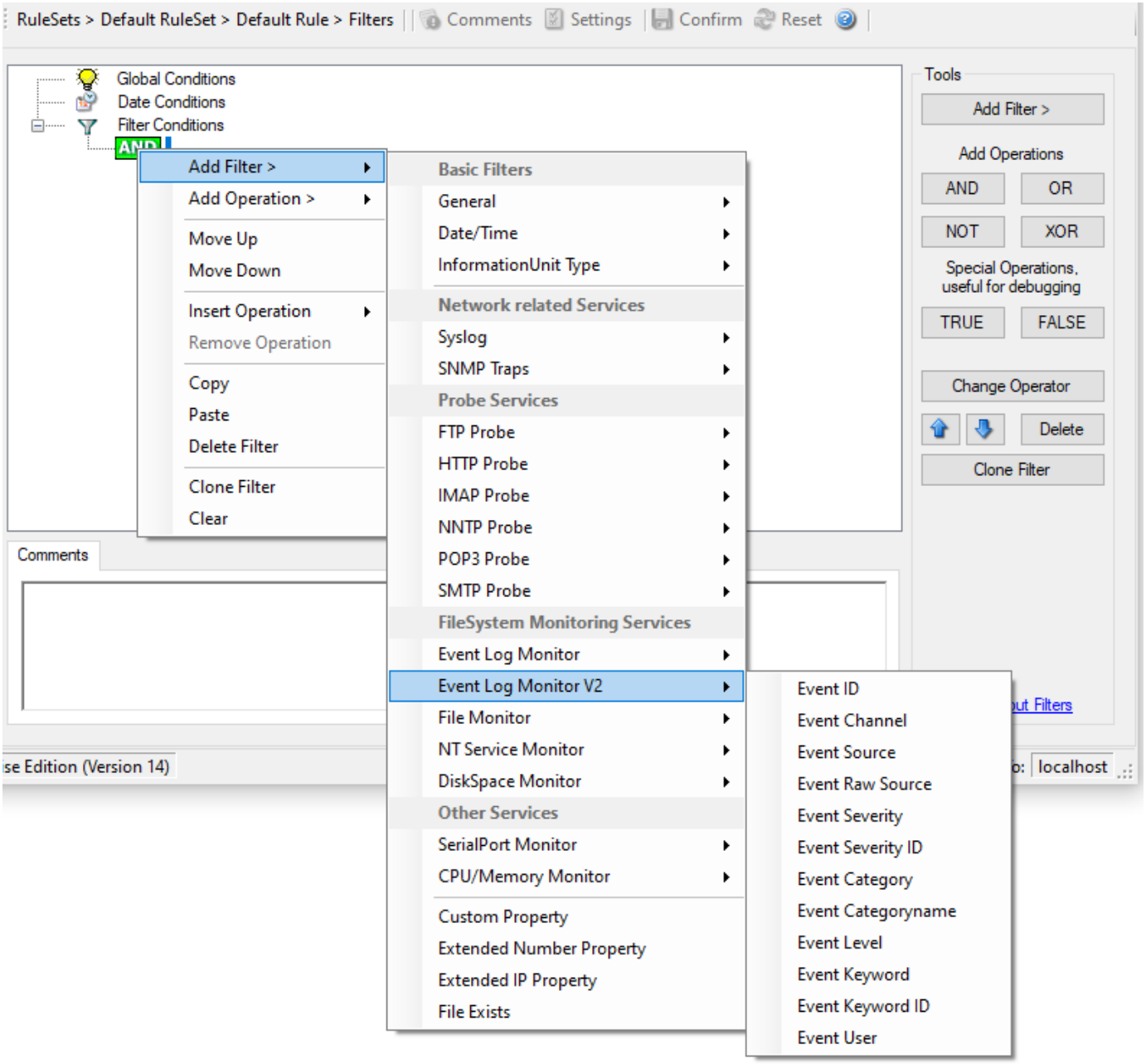
This is the event log user as specified in the Windows Event Log. If enabled, the event must have the configured event user or the rule will not match. Since it is a string value there must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event Log Monitor V2

Event Log Monitor V2 specific filters are grouped here.



- Filter Conditions - Event Log Monitor V2*

Event Channel

The channel property for event log entries, for classic Event logs they match the %nteventlogtype% property, for new event logs, they match the “Event Channel”. If enabled, the event must have the configured event type or the rule will not match. There must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event Raw Source

This contains the full internal name of the event source for new event logs, for classic event logs it contains the same value as in %sourceproc%. If enabled, the event must have the configured event source or the rule will not match. This is a string value. There must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event SeverityID

This is the internal ID of the event log level as number. This is a integer value. There must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type number.

Event Level

This is a textual representation of the event log level (which is stored as number in %severityid%). This property is automatically localized by the system. If enabled, the event must have the configured level or the rule will not match. There must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event Keyword

This is a textual representation of the event keyword. This property is automatically localized by the system. If enabled, the event must have the configured event keyword or the rule will not match. There must be an exact match. Please note that this value is case-sensitive.

This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Event KeywordID

This is the internal keyword ID as string. If enabled, the event must have the configured event keyword ID or the rule will not match. There must be an exact match. Please note that this value is case-sensitive.

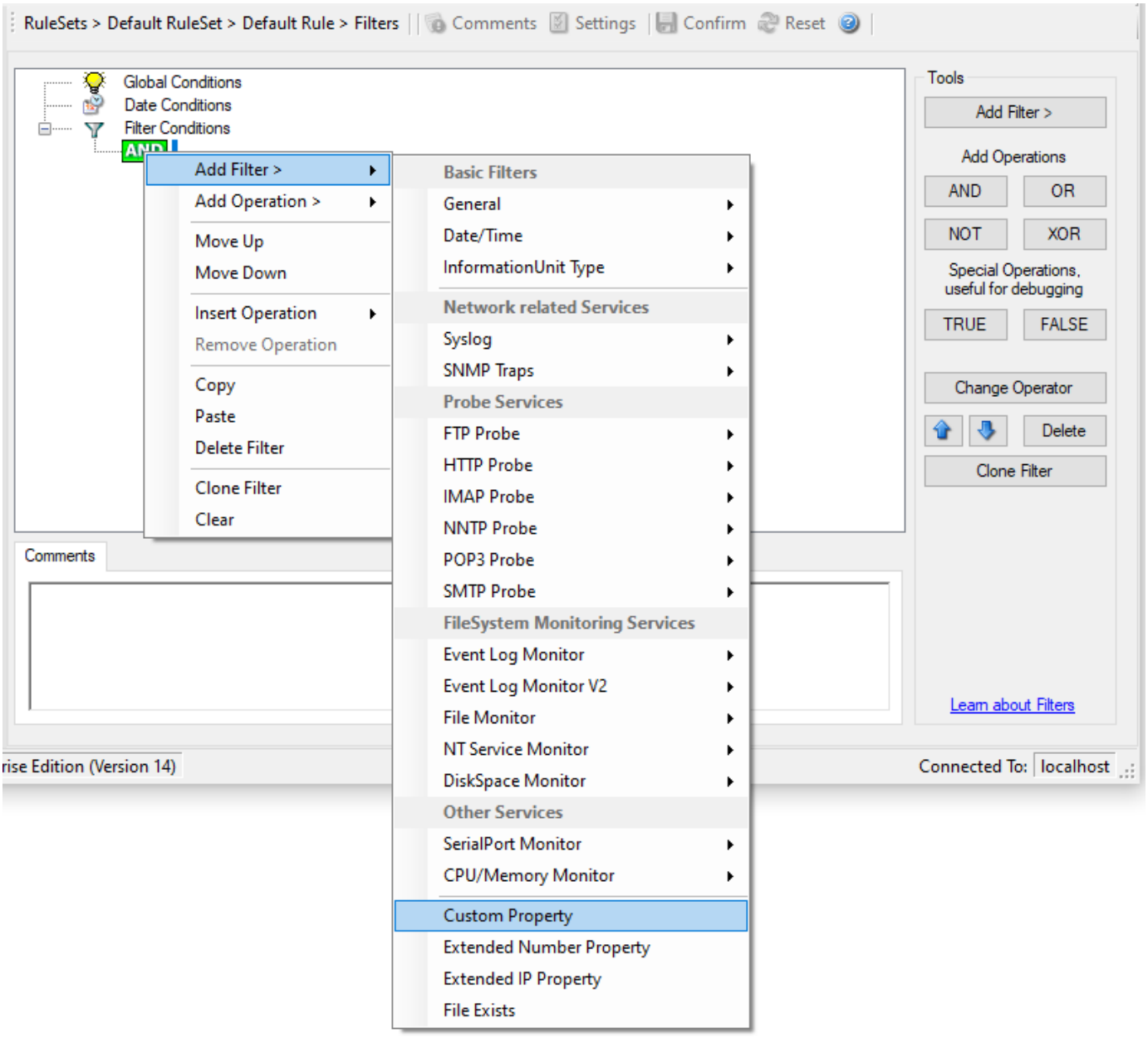
This filter condition should only be used with event log information units. If used with others, a mapped value is to be used which might not properly reflect the actual value.

This filter is of type string.

Custom properties

Custom Property

Custom Property specific filter is described here.



- Filter Conditions - Custom Property*

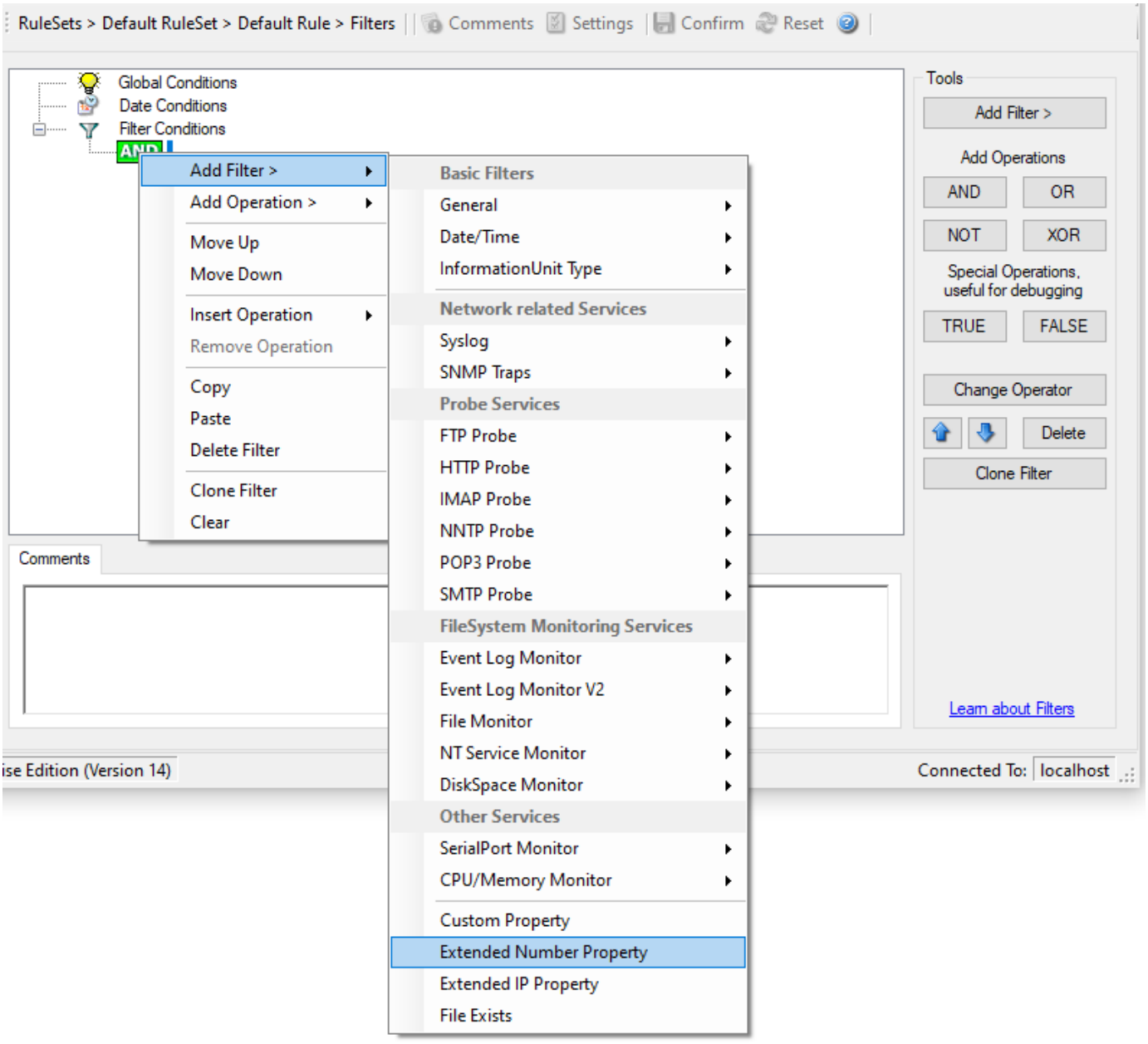
Custom Property

As the name suggests it is a “Custom Property”. Internally in MonitorWare Agent all values are stored in properties. For example the main message is stored in a property called “msg”. By using this dialog you can access properties which are dynamic (Like those from SNMP Trap Monitor when using V2 protocol).

This filter is of type string.

Extended Number Property

Extended Number Property specific filter is described here.



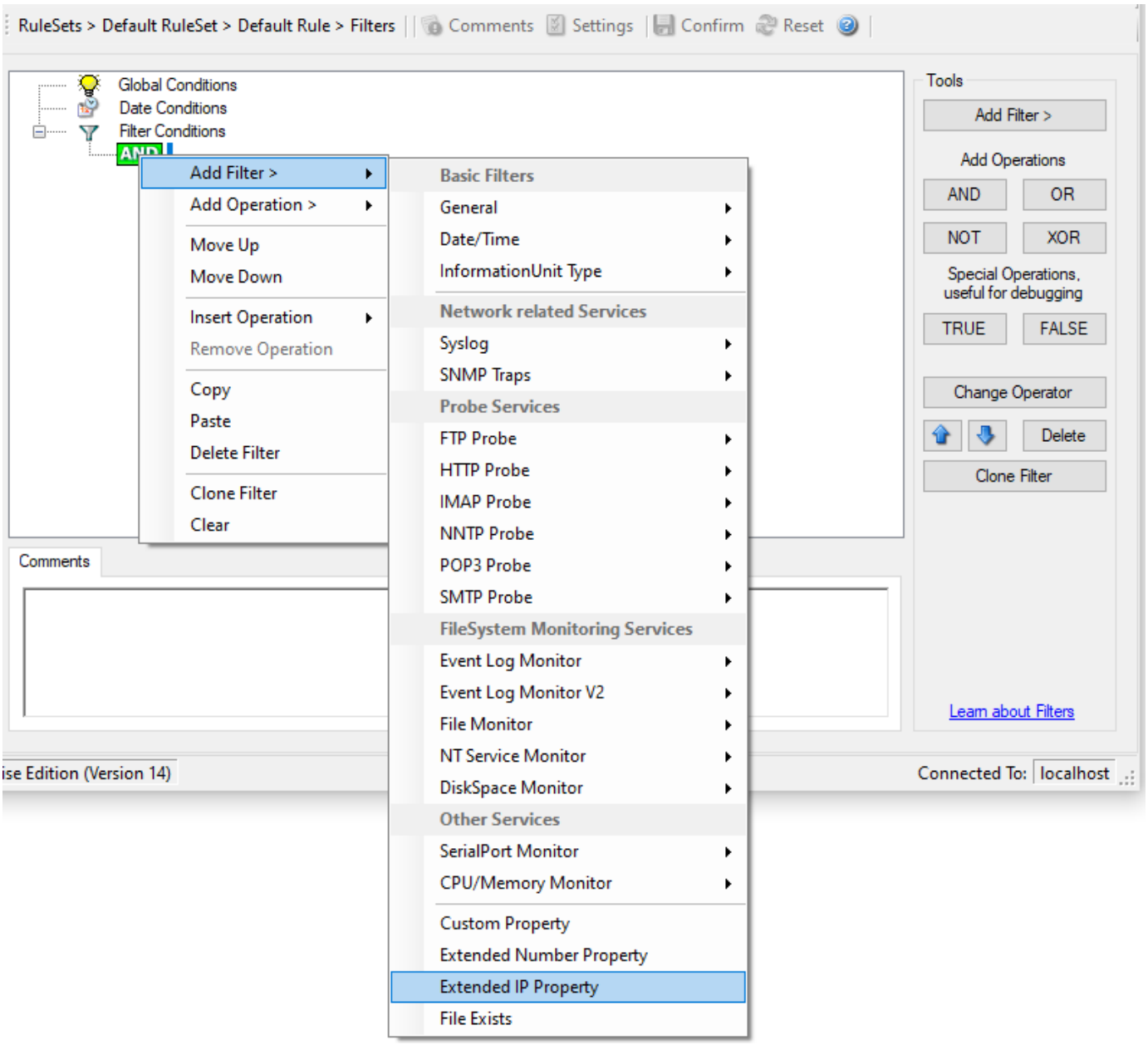
- Filter Conditions - Extended Number Property*

Extended Number Property

As the name suggests it is a “Extended Number Property”. Internally in MonitorWare Agent all values are stored in properties. For example the main message is stored in a property called “msg”. By using this dialog you can access properties which are dynamic (Like those from SNMP Trap Monitor when using V2 protocol).

This filter is of type numeric.

Extended IP Property



- Filter Conditions - Extended IP Property*

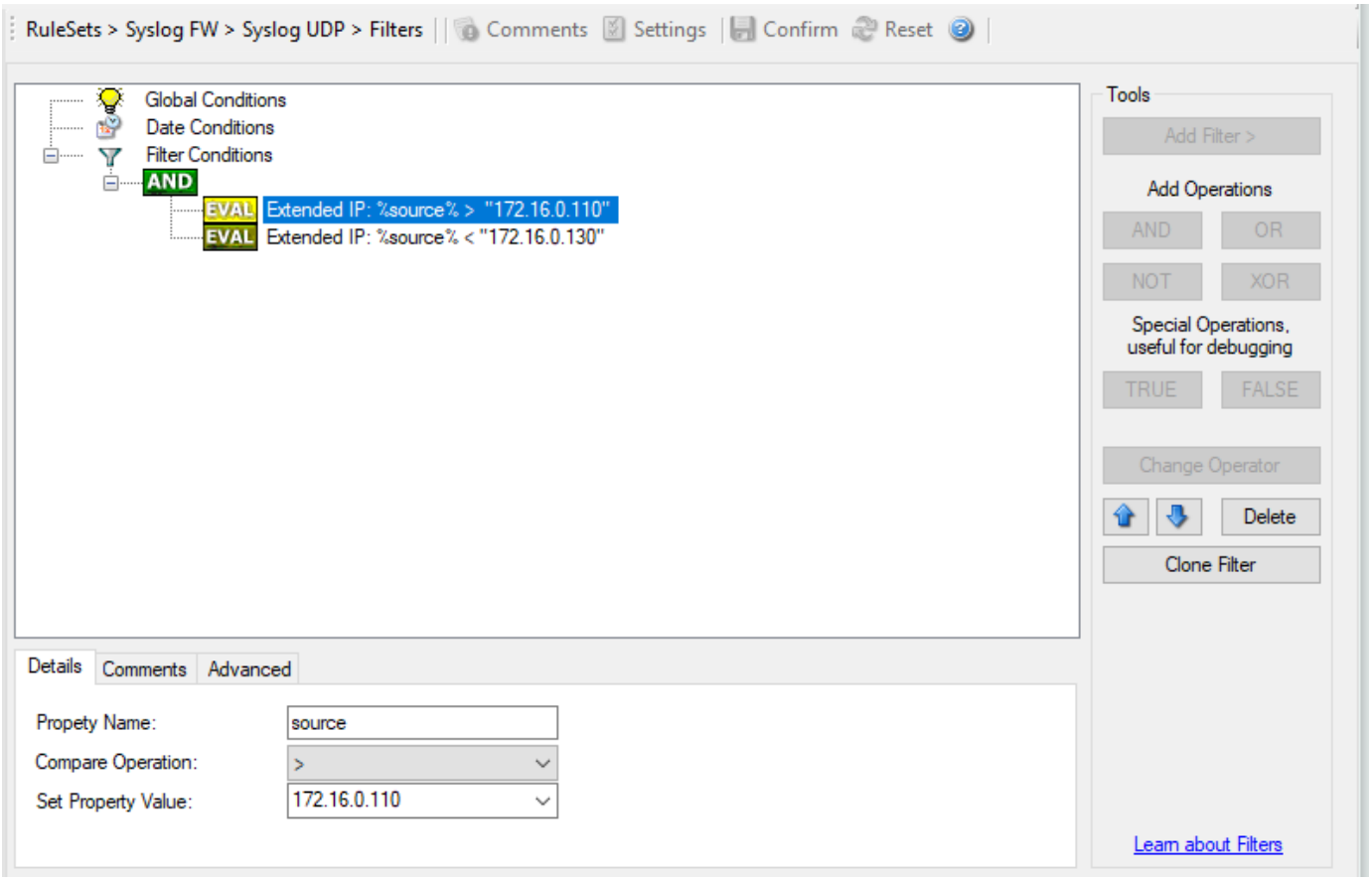
Extended IP Property filter settings

The IP Filter can basically work on any property, but we recommend to only use it on the %source% property, as we usually can be sure that this contains a valid IP Address or hostname. The IP Filter can filter against hostnames and IP Addresses, hostnames are automatically resolved using the internal DNSCache (for obvious performance reasons). If you are going to use a different or custom property, please make sure, that the data in the property is a valid IP Address.

Available compare operations for the IP Filter Type are:

Equal (=): The IP Address must match the one you configured in the Property Value field. Not Equal (!=): The IP Address must not match the one you configured in the Property Value field. Higher (>): The IP Address must be higher than the one you configured in the Property Value field. You can use IP Address Formats like: `192.168.0.10, 192.168.0, 192.168 or even 192`. It depends on what IP Ranges you are going to filter for. Lower (<): The IP Address must be lower than the one you configured in the Property Value field. You can use IP Address Formats like: `192.168.0.10, 192.168.0, 192.168 or even 192`. It depends on what IP Ranges you are going to filter for.

If you want to filter for IP Ranges, I recommend to use two filters to define the range, one filter with the “Higher (>)” compare operation and one with the “Lower (<)” compare operation. This could look like the following:

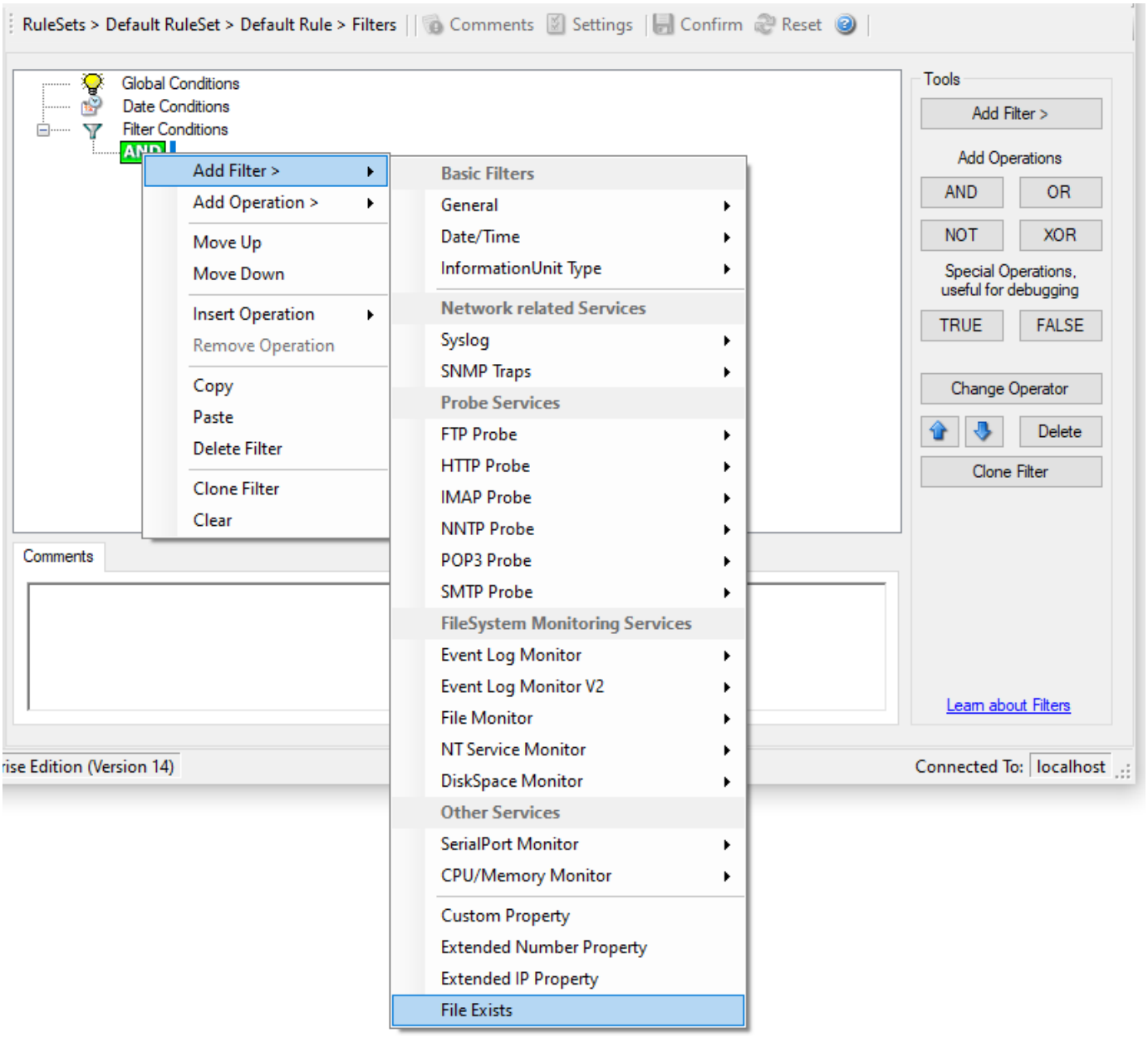


- Filter Conditions - Filtering for an IP Range*

The filter you can see here will accept all IPs which lie between `172.16.0.110` AND `172.16.0.130`. That means, that for every IP that matches these two conditions, the whole filter will evaluate to true and therefore the message will be processed. If the filter does not evaluate to true, the rule will be aborted and the message is sent to the next rule.

File Exists

Filter setting by string.



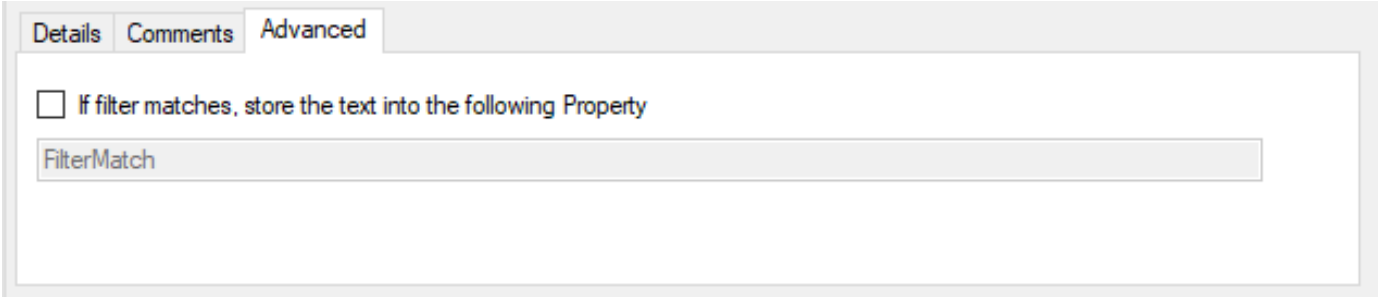
- Filter Conditions - File Exists*

File Exists

With this Filter you can simply check if a file exists or not. You can directly enter the file and its location or you can use the browse-button to find it.

Store Filter Results

How to store Filter Results is described here.



- Filter Conditions - Store Filter Results*

Store Filter Results

If a filter matches, you can now store the result of the match into a custom property.

This custom property can be used in Actions later.

Actions

Actions tell EventReporter what to do with an event after a rule matches. They store, forward, transform, or trigger follow-up behavior.

Important behavior

- A rule can contain multiple actions.
- Actions run in the order they are configured.
- Start with one simple output action during initial setup so the event path is easy to verify.

Storing actions

ODBC Database Options

Use this action to write matched events or messages to a database through ODBC.

The ODBC database action is an integration feature. It can write to the built-in Adiscon default schema or to a user-defined schema in any supported ODBC database. Use the default schema when you want the fastest supported setup or compatibility with Adiscon tools. Use custom mapping when the product needs to write into an existing database design.

Common usage patterns

- **Default schema:** Use the built-in `SystemEvents` and `SystemEventProperties` tables when you want the shortest supported setup or when downstream Adiscon tools expect the standard schema.
- **Custom schema integration:** Map event properties to an existing table with your own column names and data types.
- **Microsoft SQL Server stored procedures:** Use the call-statement option only when your SQL Server design requires a stored procedure instead of a standard `INSERT` statement.

Out of scope

This action does not design your database for you. It does not decide table layout, indexes, retention policy, reporting logic, or broader analytics architecture. For custom integration, you own the destination schema and the mapping decisions.

Before you start

- Install a supported ODBC driver on the Windows host that runs the service.
- Create an **ODBC System DSN** for the target database. User DSNs and file DSNs are not suitable for the service path.
- Verify that the database server is reachable and that the configured credentials have the required permissions.
- Decide whether the action should:
  - create and use the default Adiscon tables, or
  - write into an existing user-defined table

Minimal action path

1. Create and test an ODBC **System DSN** outside the product.
2. Add a **Write to Database** action to the relevant ruleset.
3. Configure the DSN, credentials, and connection settings.
4. Choose one of these paths:
   - keep the default schema and use **Create Database**, or
   - set the table name and field list for a custom schema
5. Save and apply the configuration.
6. Send a matching test event or message and verify that rows are inserted.

Default schema versus custom schema

Default schema

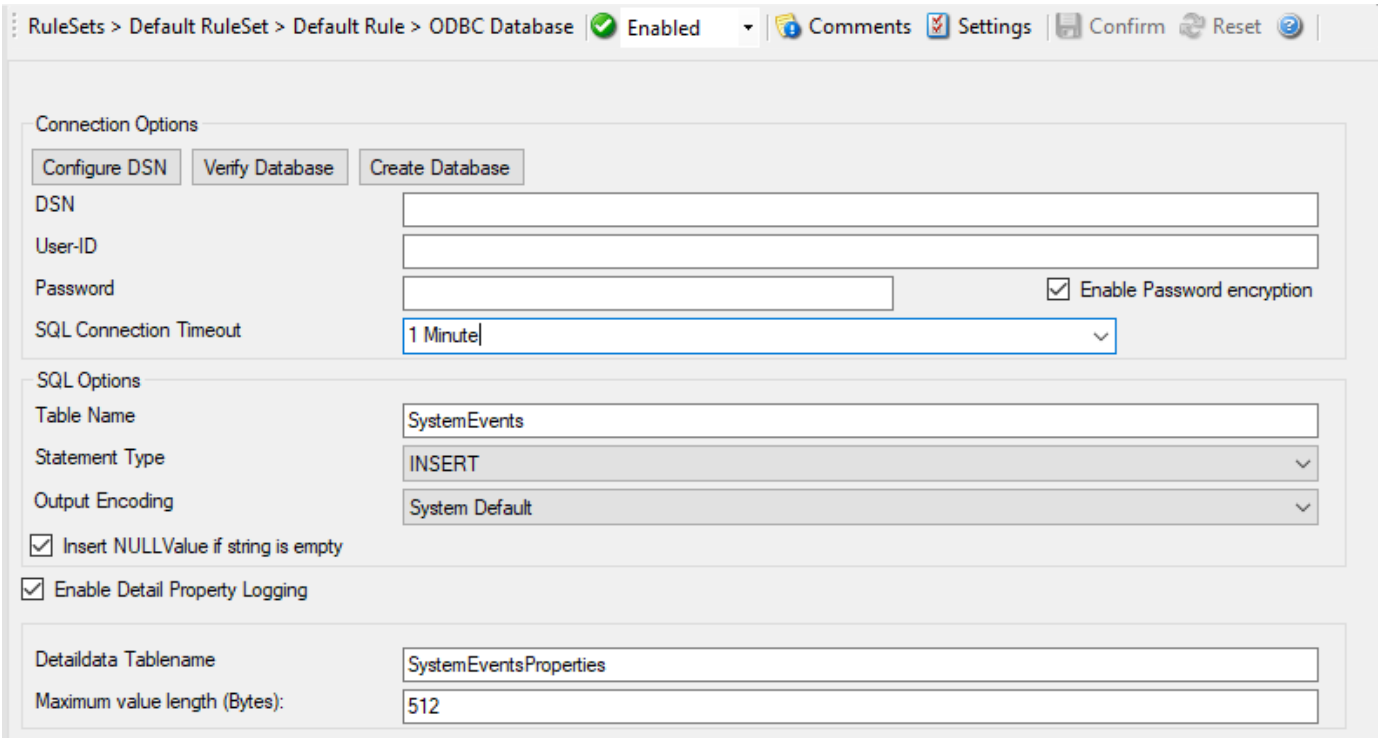
Use the default schema when you want a predictable starting point or when you need compatibility with other Adiscon components that expect the standard table layout. In this path, the action can create the default tables for you.

Custom schema integration

Use a custom schema when your organization already has a database design that WinSyslog, EventReporter, or MonitorWare Agent must write into. In this path, the action does not infer the schema. You must set the target table name and map each field deliberately.

If you diverge from the default schema, do not assume that Adiscon tools that expect the standard layout will continue to work unchanged.

Connection options



Action - ODBC Database Connection

Buttons

Configure DSN

Opens the Windows ODBC administrator so you can add, edit, or remove data sources.

Verify Database

Attempts to connect to the configured ODBC System DSN with the current settings. Use this before you save the action into production.

Create Database

Creates the default Adiscon tables in the target database. Use this only when you intentionally want the default schema.

DSN

File Configuration field:

szODBCDsn

Description:

Name of the ODBC **System DSN** used for the database connection. The DSN must already contain the correct driver and target-database connection details.

User-ID

File Configuration field:

szODBCUid

Description:

User name for database authentication, if the DSN and driver require it.

Password

File Configuration field:

szODBCPwd

Description:

Password for the configured user ID. Use an account with only the permissions needed for this action.

Enable Encryption

File Configuration field:

nODBCEnCryption

Description:

Stores the configured ODBC password encrypted instead of plaintext. Enable this unless you have a documented reason not to.

SQL Connection Timeout

File Configuration field:

nSQLConnectionTimeOut

Description:

Maximum time to wait while opening the database connection.

SQL options

Table Name

File Configuration field:

szTableName

Description:

Target table name for database writes. Keep the default `SystemEvents` when you use the built-in schema. Set it to your existing table name when integrating with a custom schema.

SQL Statement Type

File Configuration field:

nSQLStatementType

Description:

Selects whether the action uses a normal `INSERT` statement or a Microsoft SQL Server call statement for stored procedures. The call-statement path is Microsoft SQL Server specific.

Output Encoding

File Configuration field:

nOutputEncoding

Description:

Controls how string data is encoded when written. In most environments, **System Default** is the correct setting unless you have a confirmed character-set requirement.

Insert NULL Value if string is empty

Description:

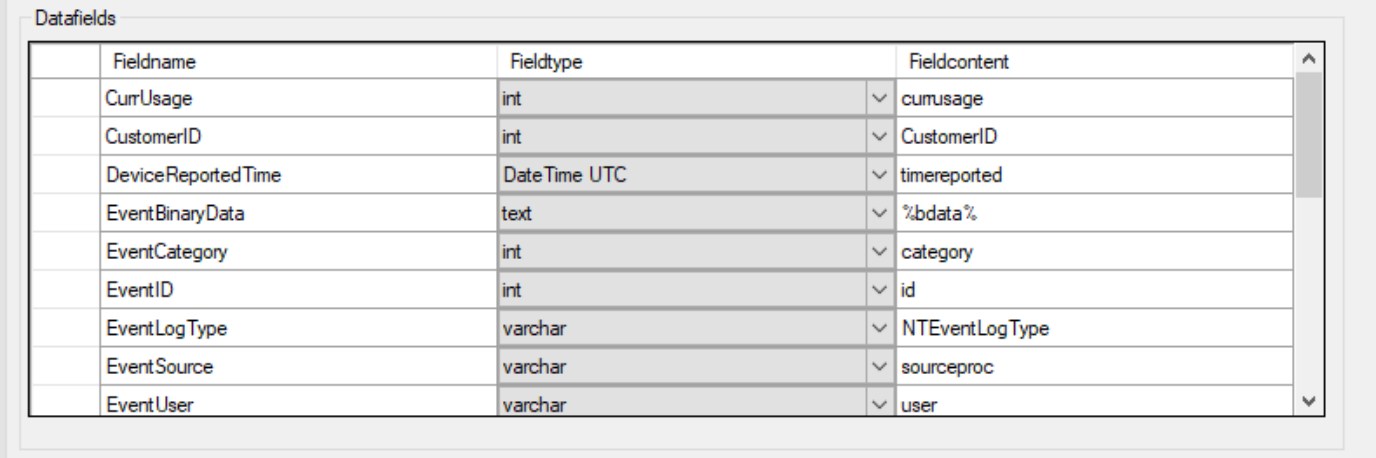
Writes `NULL` instead of an empty string for empty string properties. Use this only if your schema and downstream queries intentionally distinguish between empty text and `NULL`.

Datafields

The field list controls how event properties are written into the destination table. This is the most important part of custom integration work.

For the default schema, the built-in field list already reflects the standard Adiscon table layout. For a custom schema, keep only the rows that correspond to actual destination columns and adjust them deliberately.

For string data types, you can use the property replacer. For example, the expression `%msg:1:200%` stores only the first 200 characters of the message. For simple mappings, use the relevant event property directly.


Datafields

	Fieldname	Fieldtype	Fieldcontent
	CurrUsage	int	currusage
	CustomerID	int	CustomerID
	DeviceReportedTime	DateTime UTC	timereported
	EventBinaryData	text	%bdata%
	EventCategory	int	category
	EventID	int	id
	EventLogType	varchar	NTEventLogType
	EventSource	varchar	sourceproc
	EventUser	varchar	user

Action - ODBC Database Datafields

Fieldname

File Configuration field:

szFieldName_[n]

Description:

Database column name in the destination table.

Fieldtype

File Configuration field:

nFieldType_[n]

- 1 = varchar
- 2 = int
- 3 = text
- 4 = DateTime

Description:

Data type of the destination column. It must match both the database schema and the kind of property you are storing.

Fieldcontent

File Configuration field:

szFieldContent_[n]

Description:

Event property or property-replacer expression written into the destination column. See event properties and property access and replacer syntax.

Practical mapping guidance

For a custom syslog-oriented table, a minimal mapping often includes:

- a timestamp column populated from `timegenerated` or `timereported`

- a source column populated from `source`
- a severity column populated from `syslogpriority`
- a tag or application column populated from `syslogtag` or `syslogappname`
- a message column populated from `msg`

If a destination column is shorter than the source property, truncate or transform the value explicitly instead of hoping the driver or database will do the right thing.

Detail property logging

Enable Detail Property Logging

File Configuration field:

nPropertiesTable

Description:

Writes non-standard properties into a separate detail table. This can be useful when additional event metadata must be retained, but it also increases write volume.

Detaildata Tablename

File Configuration field:

szPropertiesTableName

Description:

Table name used for detail-property logging. In the default schema, this is typically `SystemEventProperties`.

Maximum value length (Bytes)

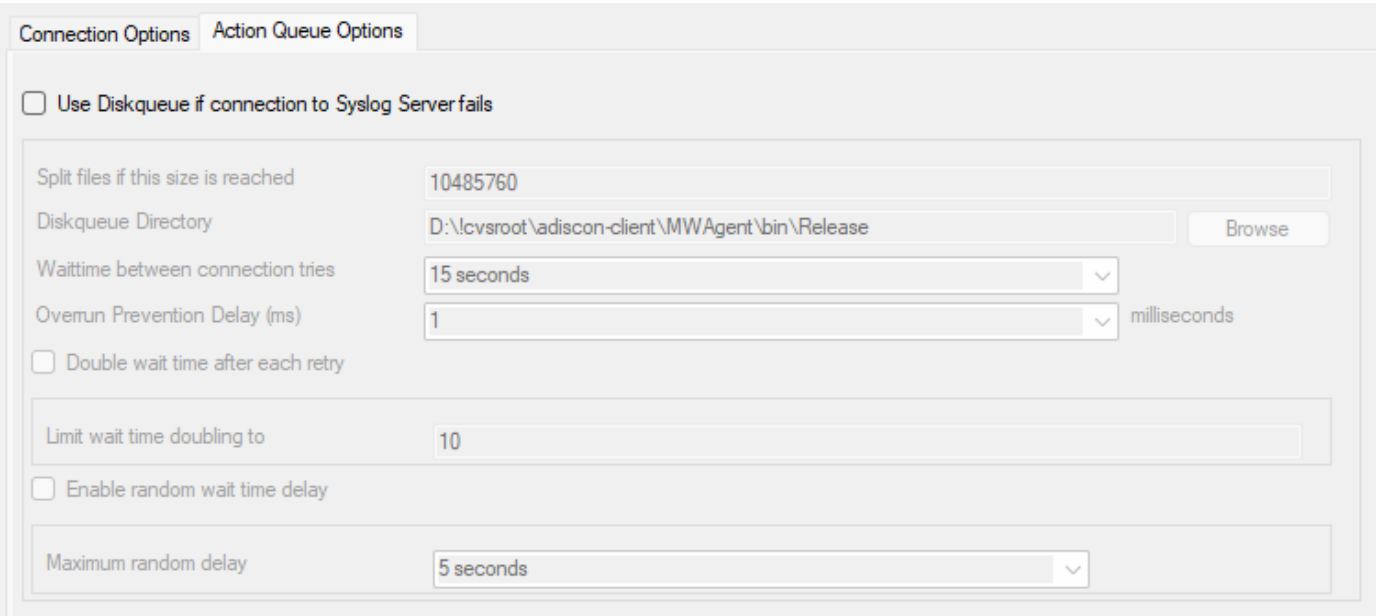
File Configuration field:

nMaxValueLength

Description:

Maximum size in bytes for values written into the detail-property table.

Action Queue Options



Action - ODBC Database Action Queue

Use Diskqueue if connection to database fails

File Configuration field:

nUseDiscQueue

Description:

Stores pending writes on disk when the database path is temporarily unavailable.

Split files if this size is reached

File Configuration field:

nDiskQueueMaxFileSize

Description:

Maximum size of each queue file in bytes before a new file is created.

Diskqueue Directory

File Configuration field:

szDiskQueueDirectory

Description:

Directory used to store queue files for pending database writes.

Waittime between connection tries

File Configuration fields:

nDiskCacheWait

Description:

Minimum wait time before the action retries the database connection after a failure.

Overrun Prevention Delay (ms)

File Configuration field:

nPreventOverrunDelay

Description:

Optional delay between replayed queue writes to avoid overwhelming the target database after recovery.

Double wait time after each retry

File Configuration field:

bCacheWaittimeDoubling

Description:

Doubles the retry wait time after each failure.

Limit wait time doubling to

File Configuration field:

nCacheWaittimeDoublingTimes

Description:

Maximum number of retry wait-time increases after repeated failures.

Enable random wait time delay

File Configuration field:

bCacheRandomDelay

Description:

Adds a randomized delay to retry timing. This can reduce synchronized retry spikes when many senders reconnect at the same time.

Maximum random delay

File Configuration field:

nCacheRandomDelayTime

Description:

Upper bound for the additional randomized retry delay.

Verification

- Use **Verify Database** before enabling production traffic.
- Send a matching test event or message after saving the action.
- Query the destination table and confirm that:
  - rows are inserted
  - values appear in the expected columns
  - data types and lengths are compatible with the schema

Common pitfalls

- Using a user DSN instead of a **System DSN**
- Leaving the default field list unchanged while targeting a custom table
- Using **Create Database** when the goal is an existing custom schema
- Mapping text properties into integer or datetime columns
- Using the SQL Server call-statement mode on non-Microsoft SQL Server targets
- Forgetting that custom schemas may break compatibility with tools that expect the default Adiscon layout

OLEDB Database Action

Use this action to write matched events or messages to a database through an OLEDB provider.

This action serves the same main use cases as ODBC Database Options, but it connects through OLEDB instead of an ODBC System DSN. It can write to the built-in Adiscon default schema or to a user-defined schema. Provider availability depends on your Windows environment and the database vendor's current OLEDB support.

When to choose OLEDB

- You already have a supported OLEDB provider for the target database.
- Your environment standardizes on OLEDB rather than ODBC.
- You need the same database-writing and field-mapping behavior but through an OLEDB connection path.

Use the ODBC action instead when your preferred or only supported driver path is ODBC.

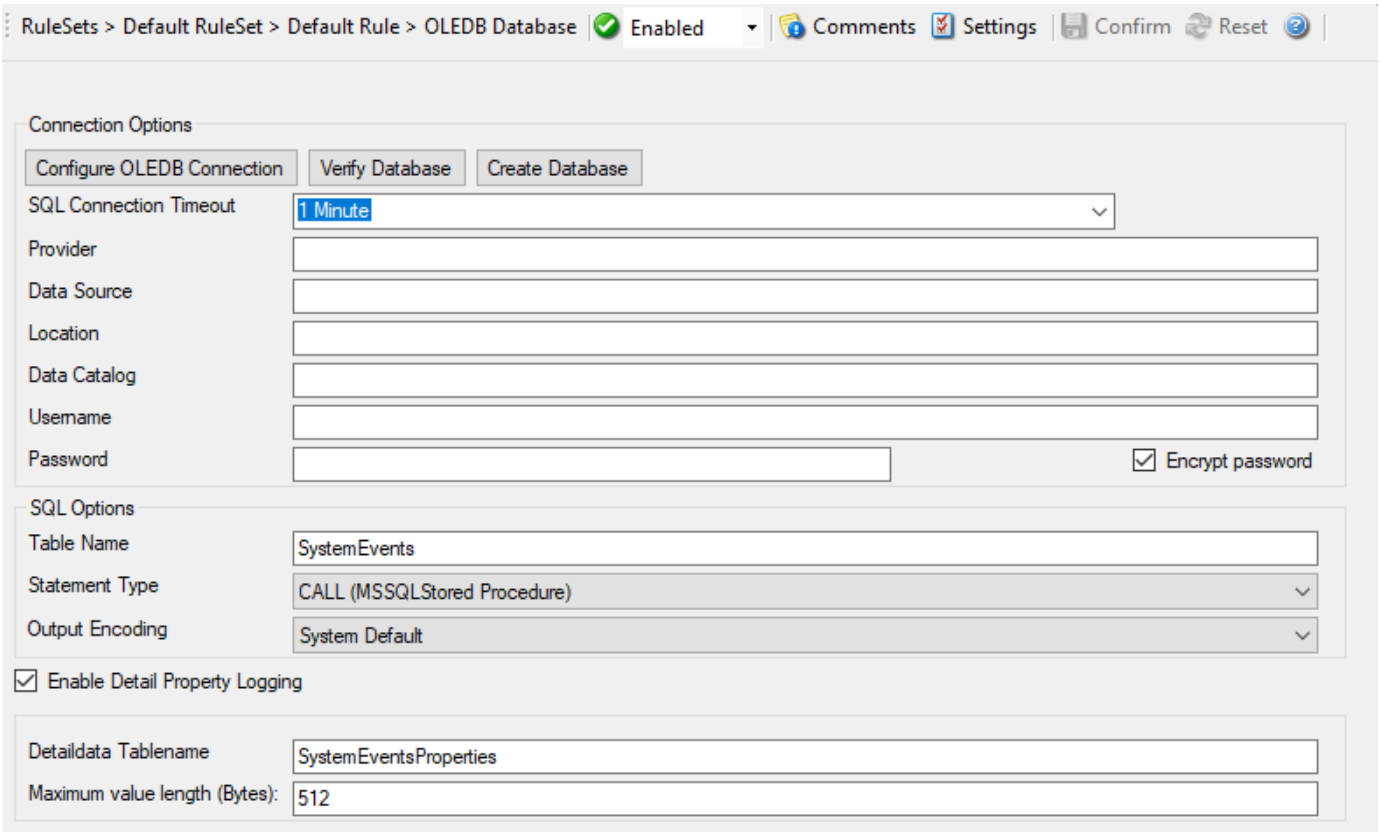
Before you start

- Verify that the required OLEDB provider is installed on the Windows host.
- Confirm the server, database, and authentication details that the provider expects.
- Decide whether you want the default Adiscon schema or an existing custom schema.
- Ensure the target account has the required database permissions.

Minimal action path

1. Configure the OLEDB connection.
2. Use **Verify Database** to test the connection.
3. Choose one of these paths:
    - use **Create Database** for the default schema, or
    - set the table name and field list for a custom schema
4. Save and apply the configuration.
5. Send a matching test event or message and verify that rows are inserted.

Connection options



Action - OLEDB Database Connection

Buttons

Configure OLEDB Connection

Starts the OLEDB configuration wizard for the provider and connection string.

Verify Database

Tests the current OLEDB connection settings.

Create Database

Creates the default Adiscon tables in the target database. Use this only when you intentionally want the default schema.

SQL Connection Timeout

File Configuration field:

nSQLConnectionTimeOut

Description:

Maximum time to wait while opening the database connection.

Provider

File Configuration field:

szProvider

Description:

OLEDB provider name. Use a provider that is actually installed and supported in your environment.

Data Source

File Configuration field:

szDataSource

Description:

Server, instance, or provider-specific data source identifier.

Location

File Configuration field:

szLocation

Description:

Optional OLEDB location setting if your provider requires it.

Data Catalog

File Configuration field:

szDataCatalog

Description:

Database name or catalog, depending on the provider.

Username

File Configuration field:

szUsername

Description:

User name for database authentication, if required by the provider.

Password

File Configuration field:

szPassword

Description:

Password for the configured user.

Encrypt password

Description:

Enable password encryption if your build exposes this option. As with ODBC, prefer encrypted storage unless you have a documented reason not to.

Table Name

File Configuration field:

szTableName

Description:

Target table name for database writes. Keep the default `SystemEvents` when you use the built-in schema. Set it to your existing table when integrating with a custom schema.

Statement Type

File Configuration field:

nSQLStatementType

Description:

Selects whether the action uses a standard `INSERT` statement or a Microsoft SQL Server call statement for stored procedures. The call-statement path is Microsoft SQL Server specific.

Output Encoding

File Configuration field:

nOutputEncoding

Description:

Controls how string data is encoded when written. In most environments, **System Default** is the correct setting unless you have a confirmed character-set requirement.

Data mapping and custom schemas

The field list works the same way as in ODBC Database Options. It controls which event properties are written to which destination columns.

For custom integration:

- set the table name to your existing table
- keep only the fields that exist in that table
- make each field name, field type, and field content match the destination schema deliberately

For string fields, you can use property-replacer expressions such as `%msg:1:200%` when you need truncation or transformation.

If you use the default schema, keep the default field list unchanged unless you understand the compatibility impact on tools that expect the standard Adiscon layout.

Detail property logging

File Configuration field:

nPropertiesTable

Description:

Writes non-standard properties into a separate detail table. This increases write volume and is usually needed only when you intentionally want those additional properties retained.

Detaildata Tablename

File Configuration field:

szPropertiesTableName

Description:

Table name used for detail-property logging. In the default schema, this is typically `SystemEventProperties`.

Maximum value length (Bytes)

File Configuration field:

nMaxValueLength

Description:

Maximum size in bytes for values written into the detail-property table.

Action Queue Options

Action - OLEDB Database Action Queue

Use Diskqueue if connection to database fails

File Configuration field:

nUseDiscQueue

Description:

Stores pending writes on disk when the database path is temporarily unavailable.

Split files if this size is reached

File Configuration field:

nDiskQueueMaxFileSize

Description:

Maximum size of each queue file in bytes before a new file is created.

Diskqueue Directory

File Configuration field:

szDiskQueueDirectory

Description:

Directory used to store queue files for pending database writes.

Waittime between connection tries

File Configuration fields:

nDiskCacheWait

Description:

Minimum wait time before the action retries the database connection after a failure.

Overrun Prevention Delay (ms)

File Configuration field:

nPreventOverrunDelay

Description:

Optional delay between replayed queue writes to avoid overwhelming the target database after recovery.

Double wait time after each retry

File Configuration field:

bCacheWaittimeDoubling

Description:

Doubles the retry wait time after each failure.

Limit wait time doubling to

File Configuration field:

nCacheWaittimeDoublingTimes

Description:

Maximum number of retry wait-time increases after repeated failures.

Enable random wait time delay

File Configuration field:

bCacheRandomDelay

Description:

Adds a randomized delay to retry timing. This can reduce synchronized retry spikes when many senders reconnect at the same time.

Maximum random delay

File Configuration field:

nCacheRandomDelayTime

Description:

Upper bound for the additional randomized retry delay.

Common pitfalls

- Assuming OLEDB is required when a supported ODBC path is simpler
- Relying on provider names or examples from older Windows environments without verifying that the provider is still installed and supported
- Using the default field list unchanged while targeting a custom table
- Expecting the action to design a custom schema automatically

File Logging Options

This configuration dialog is available both in the defaults section as well as with file logging actions.

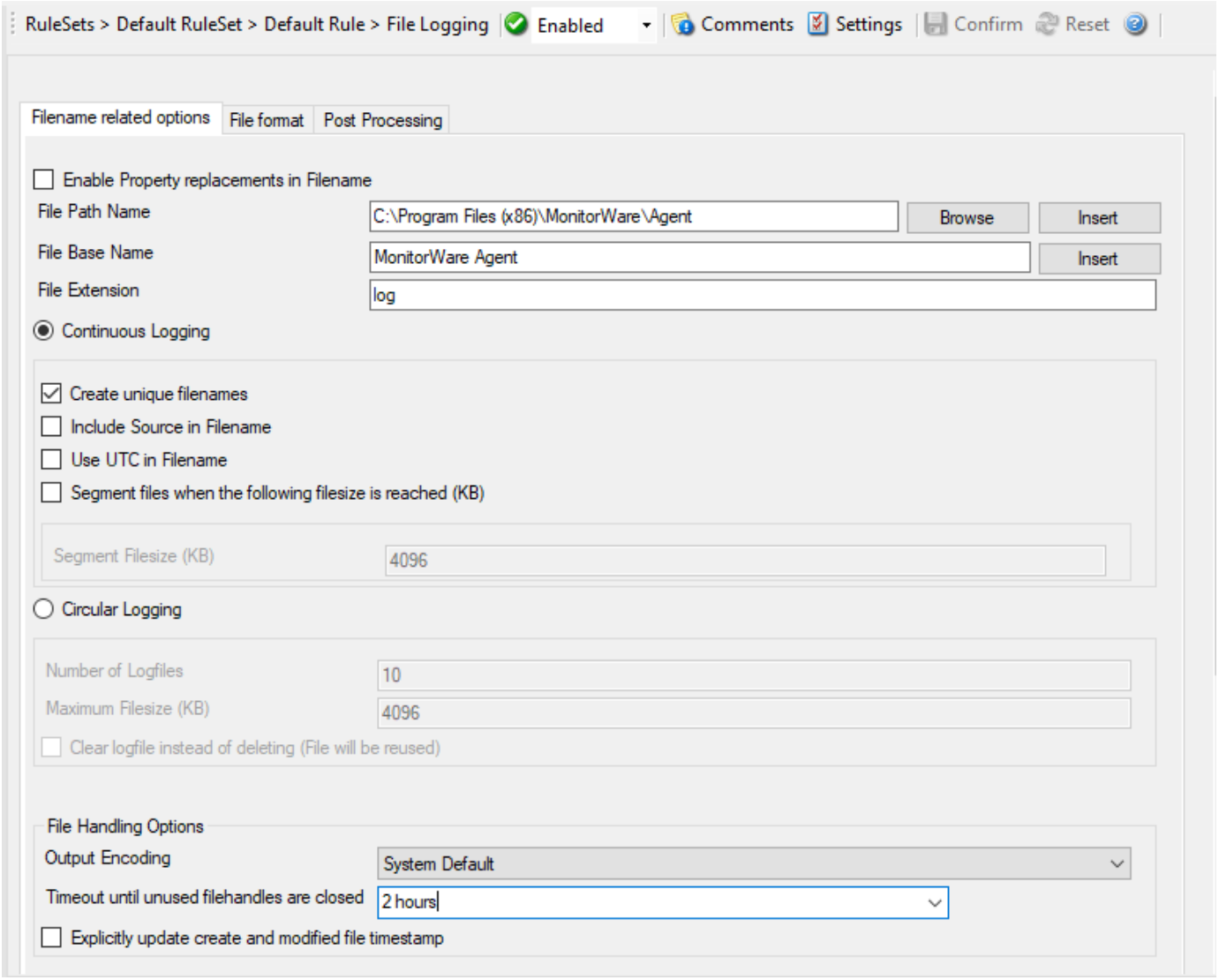
File logging is used to write text files of received messages. One file per day is written. New entries are appended to the end of the file.

File locks are released when currently no data is written. Therefore, other applications can access the files while the service is running. However, please be sure that the other applications do not place a file-lock onto it. Popular WordPad does so. In this case, the service will not be able to log any further messages (an error event is written to the Windows Event Log in this case). We recommend copying the file when accessing it at runtime - or use notepad.exe, which does not place file-locks on the files it opens.

The filename is build as follows:

`<FilePathName><FileBaseName>-year-month-day.<FileExtension>`

Parameters in the brackets can be configured via dialog shown below:



- Action - File Logging Filename related*

Enable Property replacements in Filename

File Configuration field:

nEnablePropertyFileName

Description:

By activating this option, you can use properties within the file or pathname like `%source%` and all the others. For example: File Path Name can be `F:\syslogs\%source%` File Base Name can be `IIS-%source%`

If your source is `10.0.0.1`, that writes the following file: `F:\syslogs\10.0.0.1\IIS-10.0.0.1.log`

The path `f:\syslogs\10.0.0.1` was generated because the source property was used inside the path.

Please Note that you can use ANY property inside the path and base name. event properties are described in the property replacer section.

File Path Name

File Configuration field:

szFilePath

Description:

The base path (directory) of the file. Please see above for exact placement. Default is `c:\temp`. The Insert Menu entry allows you to create “Dynamic Directories”. For example:

File Path Name can be **``F:syslogs%source%``**

event properties are described in the property replacer section.

On network paths: The File Logging action can also work on network storages. There are two ways of storing log files in a network path.

1. Direct the action to a full UNC path. In this case, make sure the system account with which the service is running is able to access the network path or the service will fail to access with a permission error. Sample path: `\Hostname\folder1\folder2\`
2. Map the UNC path to a local drive letter in Windows. In this case, the path will look like a regular local path, but actually points to a network location. This requires a workaround, which is to run a scheduled task at system startup under Local System and perform a net use specifying the user and password of the share. Else, the service will not be able to access the mapped UNC path, because the mapping usually happens for interactive sessions only.

File Base Name

File Configuration field:

szFileBaseName

Description:

The base name of the file. Please see above for exact placement. Default is “MonitorWare”. The Insert Menu entry allows you to recreate “Dynamic Base Filenames”. For example:

File Base Name can be `IIS-%source%`

File Extension

File Configuration fields:

szFileExtension

Description:

The extension to be used when writing the file. Please see above for exact placement. Default is `.log`.

Continuous Logging

Description

When enabled log files will not be overwritten, there is a single file with consistent file name. See below checkboxes to choose in which cases a new file should be created.

Create unique Filenames

File Configuration field:

nUniqueFileName

Description:

If checked, a unique file name is created for each day. This is done by adding the current date to the base name.

If left unchecked, the date is not added and as such, there is a single file with consistent file name. Some customers that have custom scripts to look at the file name use this.

Include Source in Filename

File Configuration field:

nIncludeSourceInFilename

Description:

This works together with the “Create unique Filenames” setting. If checked, the file name generation explained above is modified. The source of the Syslog message is automatically added to the file name.

This feature has been introduced because many customers would like to have separate log files for each device. While this can be achieved with multiple rules, it is much more straight forward with this single checkbox. If it is checked, the messages are automatically written to separate files and the file name includes the originating device information.

Use UTC in Filename

File Configuration field:

nUseUTCInFileName

Description:

This works together with the “Create unique Filenames” setting. If unique names are to be created then select the “Use UTC in Filename” option, in this case the file name is generated on the basis of universal coordinated time (UTC) or on local time. UTC was formerly referred to as “GMT” and is the basis of the time zone system. For example, New York, USA is 5 hours behind UTC. Therefore, if it is 12 noon in New York, the UTC time is 5pm.

When it comes to log file creation, it means that the date is computed on UTC. Taking the same example, if the “Use UTC in Filename” is checked, the log file name would roll over to the next date at 7 pm New York time. If it were unchecked, the rollover would occur exactly at midnight New York time (5 am UTC).

Using UTC for file name creation can be helpful if log files are written among different time zones and later consolidated. Using UTC ensures a consistent time notation across all log files.

Please note that this setting does affect the file name creation only. A different setting controls the dates recorded inside the file.

Segment files when the following file size is reached (KB)

File Configuration field:

nSegmentFileEnable

Description:

Files are segmented if the defined file size: Segment Filesize (KB) is reached. A sequence number is appended to the file name: _1 to _n.

Circular Logging

File Configuration field:

nCircularLogging

Description:

If enabled, log files are created and overwritten in a cycle.

Number of Log Files

File Configuration field:

nNumberOfLogfiles

Description:

Once the last log file is reached, circular logging begins and overwrites the first log file again. If set to 0, log files will not be rotated but can still be processed by Rotate Post Processing (for example compression or backup) along with the Rotate Conditions.

Maximum Filesize (KB)

File Configuration field:

nMaxFileSize

Description:

Max filesize of a log file, once this size is reached a new logfile is created.

Clear logfile instead of deleting (File will be reused)

File Configuration field:

nReUseFile

Description:

This option causes the File Action to truncate the log file instead of deleting and recreating it.

File Handling Options

Output Encoding

File Configuration field:

nOutputEncoding

Description:

This setting is most important for Asian languages. A good rule is to leave it at “System Default” unless you definitely know you need a separate encoding. “System Default” works perfectly in the far majority of cases, even on Asian (e.g. Japanese) Windows versions.

Timeout until unused filehandles are closed

File Configuration field:

nCleanFileHandlesTimeout

Description:

When dynamic filenames are used, filehandles are cached internally to avoid massive amount of File open/close operations. This timeout specifies after which time handles should be finally closed if not used anymore. Each write to a file will reset the timeout counter for the current filehandle.

Explicitly update create and modified file Timestamp

File Configuration field:

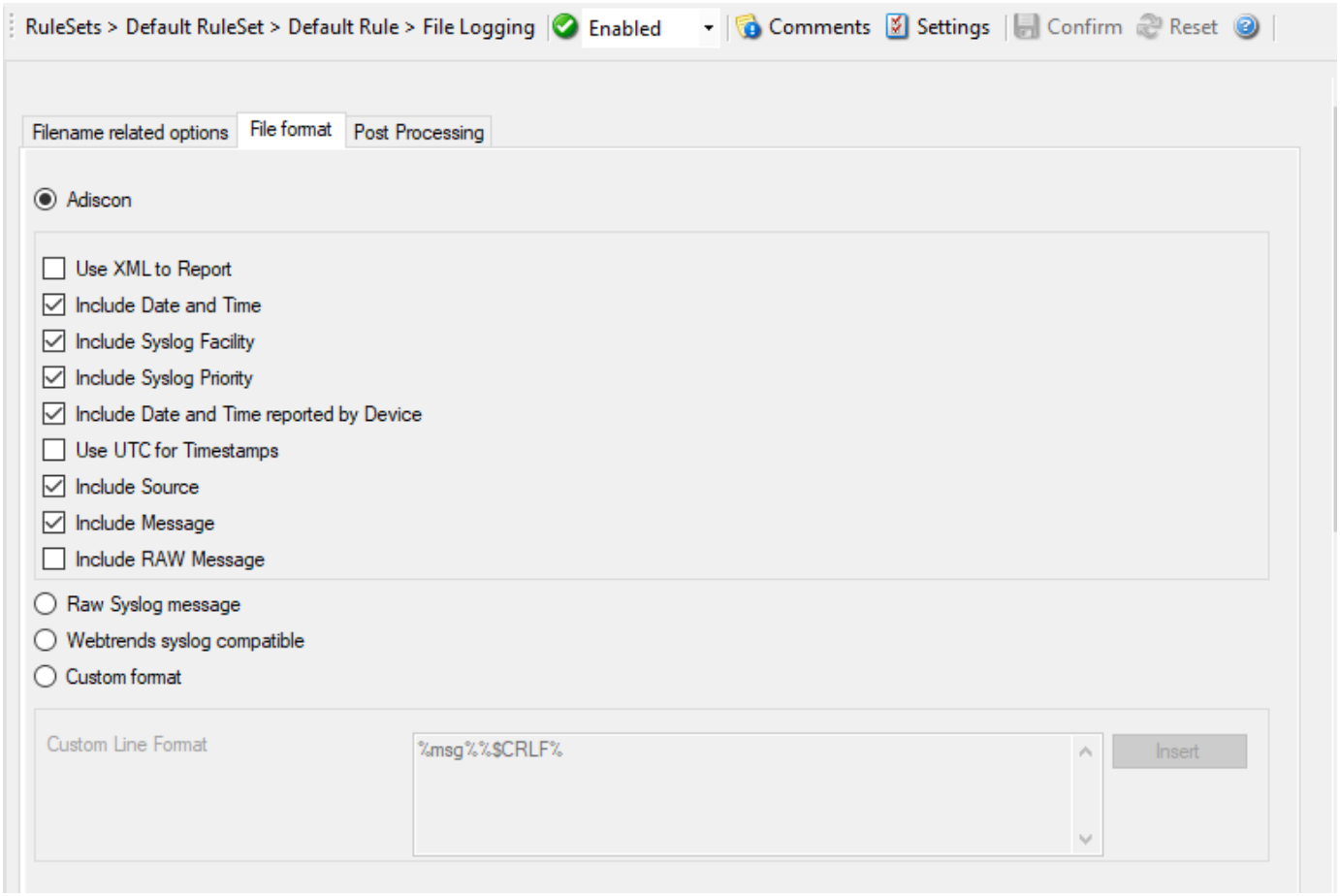
nEnableUpdateFileTime

Description:

If the checkbox is not selected the operating system updates the timestamps for creating and modifying files. In cases where the filesystem does not do this reliably, the checkbox can be selected. Now the service itself updates the timestamps for creating and modifying files.

File Format

The format in which the log file is written can be selected here. The default is “Adiscon”, which offers most options. Other formats are available to increase log file compatibility to third party applications.



- Action - File Logging File Format*

Adiscon

> **Note**
>
> Any other format besides “Adiscon Default” are fixed formats. As such, if it is selected, all other formatting options do not apply and consequently are turned off.

The following options are possible:

Use XML to Report

File Configuration field:

nUseXMLtoReport

Description:

If checked, the message part includes a complete XML-formatted information record. It includes additional information like timestamps, syslog facility and priority, and others in an easy to parse format. If XML output format is selected, you might consider turning all other information fields off, as they are already included in the XML stream. However, this is not a requirement.

Use UTC for Timestamps

File Configuration field:

nUseUTCForTimestamps

Description:

Please see the definition of utc above at “Use UTC in Filename”. This setting is very similar. If checked, all time stamps are written in UTC. If unchecked, local time is used instead. Again, UTC is useful if logs written in multiple time zones are to be consolidated.

Include <Fieldname>

File Configuration field:

- nFileDateTime
- nFileFacility
- nFilePriority
- nFileDateTimeReported
- nFileSource
- nIncludeMessage
- nIncludeRAWMessage

Description:

The various “include” settings controls are used to specify the fields which are to be written to the log file. All fields except the message part itself are optional. If a field is checked, it is written to the log file. If unchecked, it will not be written. All fields are comma-delimited.

Please note the difference between the “Date and Time” and “Date and Time reported by Device”. Both are timestamps. Either both are written in local time or utc based on the “Use UTC for Timestamps” check box. However, “Date and Time” is the time when the product received the message. Therefore, it is always a consistent value.

In contrast, the “Date and Time Reported by Device” is a timestamp taken from the actual message. As such, it is dependent on the reporting device clock, which might be off. In addition, in the case of Syslog messages, there is no time zone information within the device reported timestamp. As such, if devices from multiple time zones are reporting, the timestamp information is not consistent. This is due to Syslog design as of rfc 3164. The Syslog server can be configured to ignore the RFC in this case and provide a consistent time stamp. However, from the view of the log file writer, the “Date and Time Reported by Device” might not be as trustworthy as the “Date and Time” field. Nevertheless, it might also be more useful than the former one. This is the reason both timestamps are present and can individually be selected.

The “Include Message” and “Include RAW Message” fields allow customizing the message part that is being written. The raw message is the message as – totally unmodified, was received. This might be useful if a third party application is expecting raw Syslog entries. The message itself is just that part of the Syslog message that is being parsed as message. That is without e.g. host information or a tag value. Please note that we recommend selecting only one of these options, as otherwise two message fields are written. Similarly, if none is selected no message is written at all. Please note that we support these configurations, too – there might be a legitimate need for them.

Raw Syslog message

The “Raw Syslog message” format writes raw Syslog format to the log file. That is, each line contains the Syslog message as of RFC 3164. No specific field processing or information adding is done. Some third party applications require that format.

Webtrends syslog compatible

The “WebTrends Syslog compatible” mimics the format that WebTrends applications expect. Please note that we only mimic the log file format. It is still the job of the reporting device (most notable firewall) to generate the correct WebTrends WELF format. The “WebTrends” format is supported because many customers would like to use product enhanced features while still having the ability to work with WebTrends.

Custom format

The “Custom format” allows you to customize formats to increase log file compatibility for third party applications. When you choose this option then Custom line format is enabled.

Custom Line Format

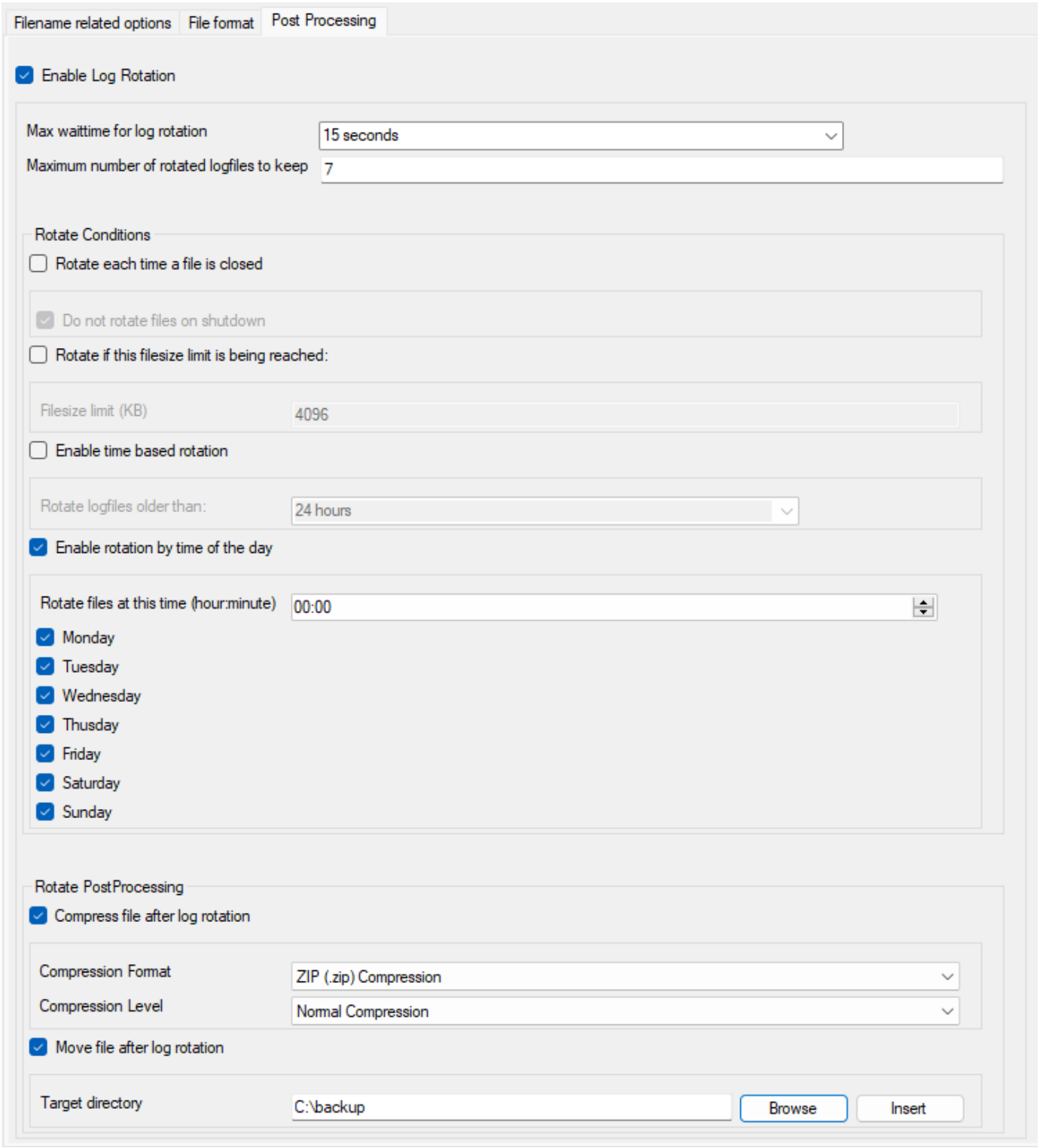
File Configuration field:

szLineFormat

Description:

Custom Line Format enables you to fully customize the output for the log file. The Insert Menu entry provides further options and they only work in custom line format. Default value is `%msg%%$CRLF%`.

Post Processing



- Action - File Logging Post Processing*

Enable Log Rotation

File Configuration field:

nLogEnableRotate

Description:

Enables the log rotation post-processing subsystem for this File Action.

Maximum wait time for log rotation

File Configuration field:

nLogRotateMaxWait

Description:

Maximum Wait time when log rotation is processed within the Queue Engine.

Maximum number of rotated log files to keep

File Configuration field:

nLogRotateMaxFiles

Description:

Maximum number of rotated log file generations to keep. If set to 0, log files will not be renamed into numbered generations but can still be processed by Rotate Post Processing (for example compression or backup) along with the Rotate Conditions.

Rotate Conditions

Rotate each time a file is closed

File Configuration field:

nLogRotateOnClose

Description:

When a file is closed (Timeout for example), log rotation will be done.

Do not rotate files on Shutdown

File Configuration field:

nLogDoNotRotateOnShutdown

Description:

Do not rotate log files if service is stopped even with “Rotate each time a file is closed” enabled.

Detached log rotation work that was already queued or already active in the background worker is still recovered on the next startup. This setting only suppresses starting an additional rotate-on-close operation because the service is shutting down.

Rotate if this filesize limit is being reached

File Configuration field:

nLogRotateOnSizeLimit

Description:

Enable log rotation if a configured file size is reached.

Filesize limit (KB)

File Configuration field:

nLogRotateSizeLimit

Description:

The actual file size in KB for “Rotate if this filesize limit is being reached”.

Enable time based rotation

File Configuration field:

nLogEnableRotateTimeout

Description:

Enable time based log rotation.

Rotate log files older than

File Configuration field:

nLogRotateTimeout

Description:

Sets the maximum file age before a logfile is being rotated when “Enable time based rotation” is enabled.

Enable rotation by time of the day

File Configuration field:

nLogEnableRotateTimeOfTheDay

Description:

Rotate this file at this time (hour:minute) and the checked day/days.

Rotate PostProcessing

Compress File After log rotation

File Configuration field:

nLogZipAfterRotate

Description:

Enable file compression after log rotation.

Compression runs in the asynchronous log rotation worker after the live file has already been detached. If graceful shutdown happens while compression is still running, the unfinished detached work is resumed on the next startup instead of being abandoned.

Compression Format

File Configuration field:

nLogZipAfterRotateFormat

Description:

It is possible to compress to ZIP or GZIP format.

Compression Level

File Configuration field:

nLogZipCompressionLevel

Description:

There are different levels that can be selected:

- Best Speed
- Low Compression
- Normal Compression

- Best Compression

Move file after log rotation

File Configuration field:

nLogMoveAfterRotate

Description:

Move logfile after rotation & compression.

Move-after-rotate uses the same asynchronous recovery model as compression. If shutdown reaches the configured worker wait timeout before the move finishes, the detached work is resumed on the next startup.

Target directory

File Configuration field:

szLogMoveAfterRotatePath

Description:

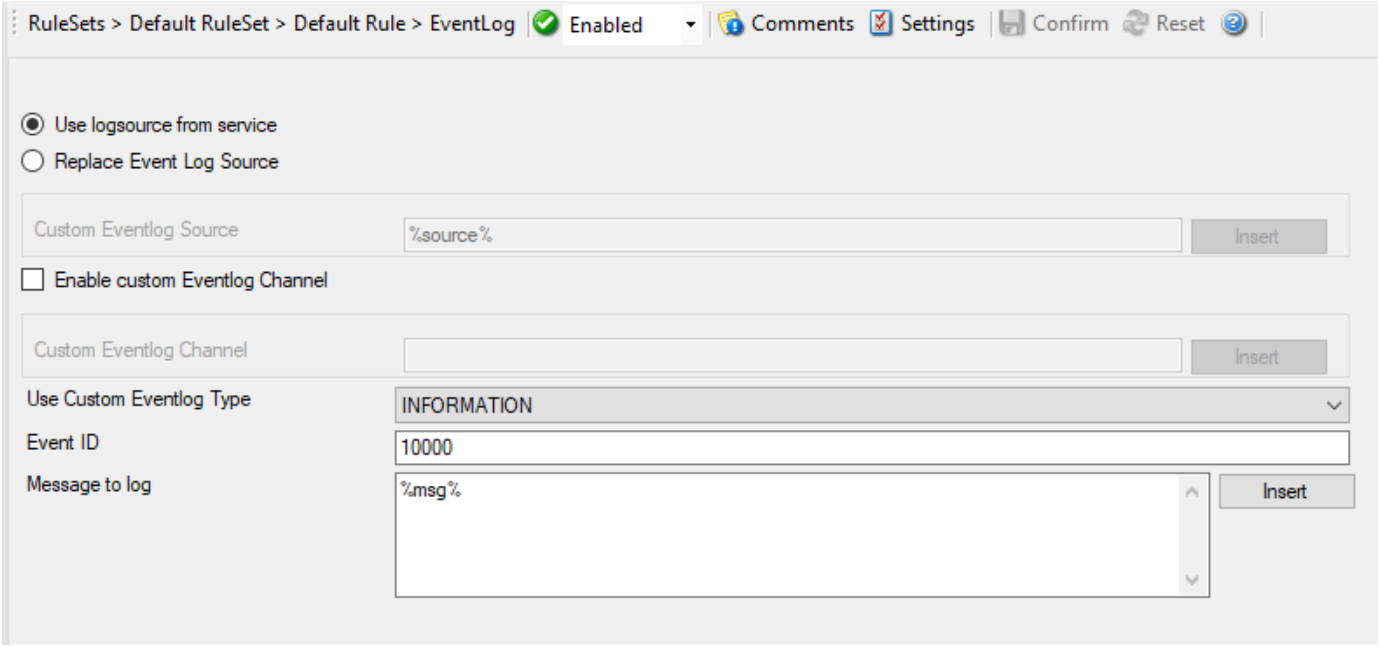
Location where to move the logfile after rotation & compression.

For an overview of shutdown, restart, reload, and dynamic-filename guarantees, see **`../shared/faq/log-rotation-guarantees`**.

Forwarding actions

Event Log Options

This tab is used to configure the logging to the Windows Event Log. It is primarily included for legacy purposes.



- Action - EventLog*

Use logsource from service

File Configuration field:

bUseCustomEventLog = 0

Description:

Takes the service name as logsource for the log entry. This option is enabled by default.

Replace Event Log Source

File Configuration field:

bUseCustomEventLog = 1

Description:

If checked, a special mapping mechanism is activated. In this mode, the Windows event source is set to the IP address of the system sending the Syslog message. In addition, the ID is set to syslog facility. This mode helps to quickly gather information about the system state in Windows event viewer.

However, this mode has its drawbacks. Effectively, we are writing invalid event source information to the event log. This does not harm any application, but Windows event viewer will try to locate the matching message libraries. Of course, this is impossible. As such, event viewer will warn the user that the message library could not be found. Nevertheless, it will display the complete logged message. This happens only in detail view.

Users should fully understand the implications of this mapping mechanism for their environment before turning this option on.

Custom Event Log Source

File Configuration field:

szCustomSource

Description:

EventSource is now fully configurable with all possibilities the property engine gives you. Please note that content of this field can be configured. event properties are described in the property replacer section.

Enable custom Eventlog Channel

File Configuration field:

bUseCustomEventLog

Description:

If enabled, a custom event log channel will be used instead of application.

Custom Eventlog Channel

File Configuration field:

szCustomEventLog

Description:

The custom Eventlog channel to be used instead of application. Will be automatically created if the channel does not exist.

Use Custom Eventlog Type

File Configuration field:

nEventType

- 0 = EVENTLOG_SUCCESS (Information event)
- 1 = EVENTLOG_ERROR_TYPE (Error event)
- 2 = EVENTLOG_WARNING_TYPE (Warning event)
- 4 = EVENTLOG_INFORMATION_TYPE (Information event)
- 8 = EVENTLOG_AUDIT_SUCCESS (Success Audit event)
- 16 = EVENTLOG_AUDIT_FAILURE (Failure Audit event)

Description:

The type – or severity – this log entry is written with. Select from the available Windows system values.

EventID

File Configuration field:

nEventID

Description:

The ID to be used when writing to the event log. Different IDs can be used to provide other processes with a consistent interface to specific messages. WinSyslog does not restrict the IDs that can be used. However, if an ID is written that is not registered with the operating system, Windows event viewer places an error message pointing this out before the actual message text. To avoid this text, event IDs 10,000 to 10,100 have been registered with the OS. We highly recommend that these IDs should be used for all custom messages. IDs below 10,000 should not be used as they might potentially interfere with events generated by MonitorWare Agent 3.0 itself.

Message to Log

File Configuration field:

szMessagecontent

Description:

It is the message which will be logged into the Windows Event Log. It is fully configurable what is logged into the Eventlog.

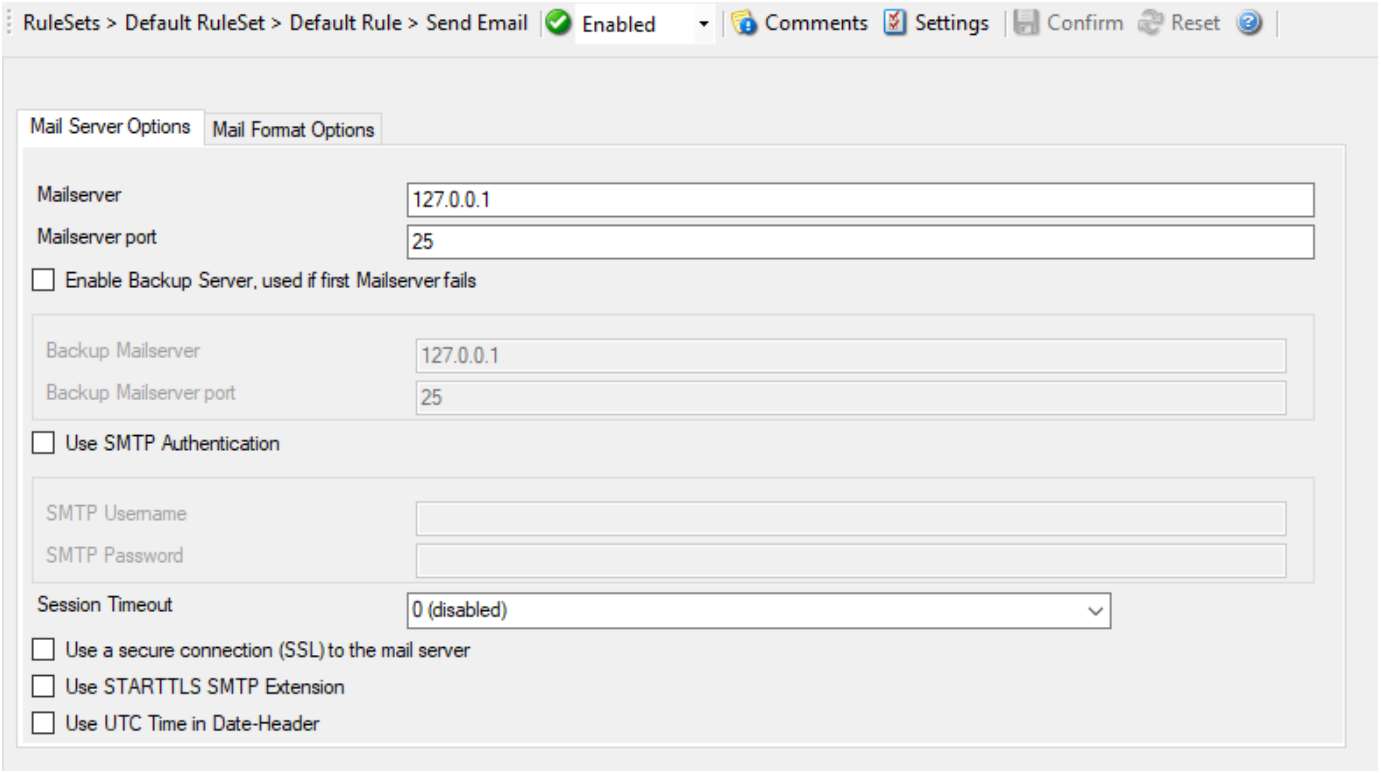
Insert Menu entry allows you to add replacement characters e.g. ``%msg%`` - you can write the actual message of an event into the Windows Event Log.

Please note that the message content of the message field can be configured. event properties are described in the property replacer section.

Send Email

This tab is used to configure mail (SMTP) parameters. These are the basic parameters for email forwarding. They need to be configured correctly, if mail message should be sent by the service.

Mail Server Options



- Action - Send Email - Mail Server Options*

Mailserver

File Configuration field:

szSMTPServer

Description:

This is the Name or IP address of the mail server to be used for forwarding messages. You can either use an IPv4, an IPv6 Address, or a Hostname that resolves to an IPv4 or IPv6 Address. Please note that this server must be able to relay messages if the recipient is not hosted at this server. Be sure to contact your mail server's administrator if in doubt on this issue.

The service expects to talk to a standard SMTP mail server. Message relaying to the final destination must be permitted.

Mailserver port

File Configuration field:

nSMTPPort

Description:

Port the mail server is to be contacted at. Usually, this is 25. It might, however, be changed in your system. Then, specify the port your mail server uses. If in doubt, try the default of 25 - or contact your mail server administrator.

Enable Backup Server, used if first Mailserver fails

File Configuration field:

nEnableBackupServer

Description:

When enabled, you can configure a second Mailserver that will be used if the regular Mailserver is not available/accessible.

Backup Mailserver

File Configuration field:

szSMTPServerBackup

Description:

In case that the connection to the main configured mail server cannot be established, the backup mail server is tried. Note that an error is only generated, if the connection to the backup server fails as well.

Backup Mailserver port

File Configuration field:

nSMTPPortBackup

Description:

Port the mail server is to be contacted at. Usually, this is 25. It might, however, be changed in your system. Then, specify the port your mail server uses. If in doubt, try the default of 25 - or contact your mail server administrator.

Use SMTP Authentication

File Configuration field:

nUseSMTPAuth

Description:

Check this box if your server requires SMTP authentication. To fight SPAM, more and more server operators allow relaying only for authenticated users. It might even happen that an existing account does no longer work because the server has been reconfigured to disallow anonymous posting.

If your server requires (or supports) SMTP authentication, check this box and enter your User ID and password in the boxes below. The exact values will be provided by your server operator – if in doubt, please ask the mail server support.

If the mail server does not support authentication, leave this box unchecked.

We recommend using authentication if it is available. Even when the current server configuration allows unauthenticated relay, this potentially will change in the future (as the SPAM problem grows). If you already use authentication, such a server configuration change will not affect you. Otherwise, it will disrupt mail service.

Session Timeout

File Configuration field:

nTimeoutValue

Description:

This option controls if multiple rapidly incoming messages should be combined to a single email message. The SMTP session with the server is held open for the specified timeout period. Please note that the period is specified in milliseconds, not seconds.

If a new event arrives within the specified timeout period, that event will be included in the same email message as the previous one. Then, the timeout is re-started. As such, any events coming in within successive timeout periods will be combined in a single mail.

This is most appropriate when large burst of messages are expected and these should be combined in few mail messages. Otherwise, multiple mail messages can easily overflow the administrator’s mailbox.

The session timeout is user configurable between 1 and 2147483647 milliseconds (32bit integer) or different pre-set values. Larger values are not supported as they probably affect the SMTP server performance and can lead to unpredictable results.

The session timeout of zero milliseconds has a special meaning: if it is selected, every event will be sent in a separate message, no matter how fast two messages occur after each other.

Use a secure connection (SSL) to the mail server

File Configuration field:

nUseSSL

Description:

This option enables SSL-secured traffic to the mail server. Please note, that this only works, if the receiving mail server supports SSL-secured transmission of emails.

Use STARTTLS SMTP Extension

File Configuration field:

nUseUTCTimeStamp

Description:

Some Email Readers do not support UTC time in date-headers. Therefore here is a switch to turn the UTC time on or off.

Use UTC Time in Date-Header

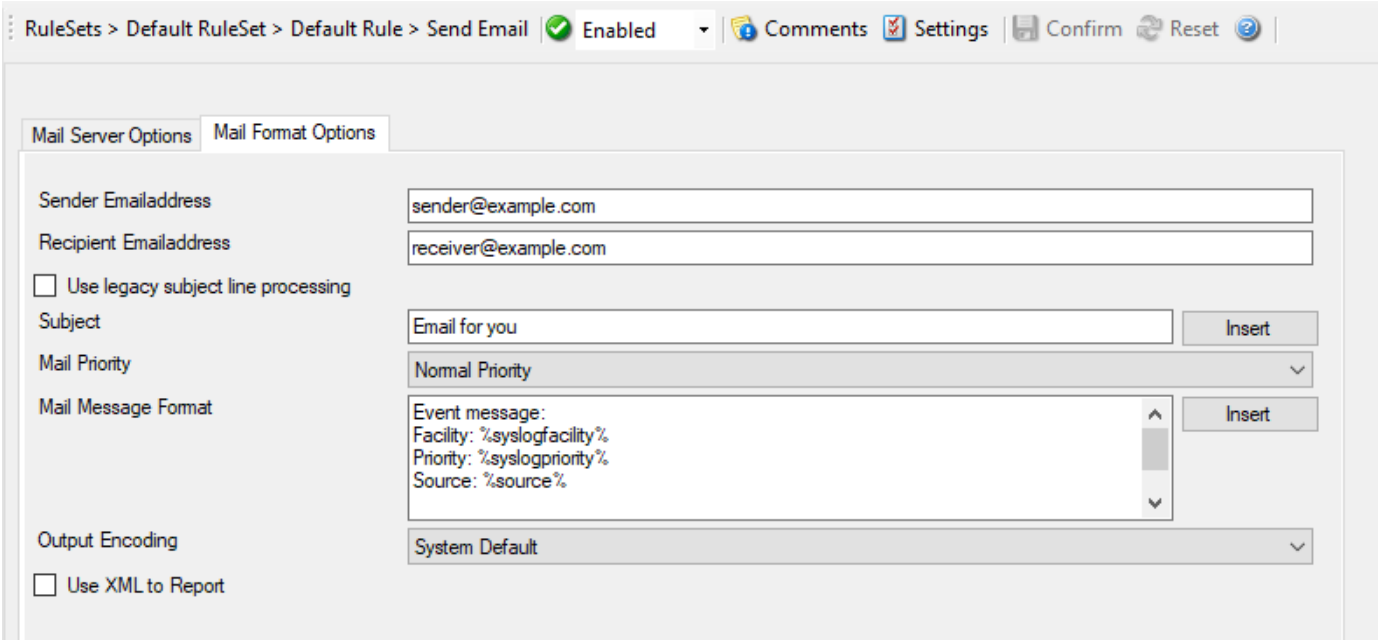
File Configuration field:

nUseUTCTimeStamp

Description:

Some Email Readers do not support UTC time in date-headers. Therefore here is a switch to turn the UTC time on or off.

Mail Format Options



- Action - Send Email - Mail Format Options*

Sender email address

File Configuration field:

szSMTPSender

Description:

Email address used as the sender address for outgoing messages. In order for your SMTP server to accept it, it probably must be a valid address.

Recipient email address

File Configuration field:

szSMTPRecipient

Description:

The recipient emails are addressed to. To send a message to multiple recipients, enter all recipient's email addresses in this field. Separate addressed by spaces, semicolons, or commas (`e.g. "receiver1@example.com, receiver2@example.com"`). Alternatively, you can use a single email address and define a distribution list in your mail software. The distribution list approach is best if the recipients frequently change or there is a large number of them. Multiple recipients are also supported. They can be delimited by space, comma, or semicolon.

Use legacy subject line processing

File Configuration field:

nUseLegacySubjectProcessing

Description:

This checkbox specifies which type of subject line processing will be done. If it is checked, the old-style processing using single character replacement sequences is applied. If it is left unchecked, the far more powerful event property based method is used.

In legacy mode, the following replacement characters are recognized inside the subject line:

`%s` IP address or name (depending on the "resolve hostnames" setting) of the source system that sent the message.

`%f` Numeric facility code of the received message

`%p` Numeric priority code of the received message

`%m` the message itself. Please note: this is the complete message text and can be rather lengthy. As such, it is most probably subject to truncation. If that occurs, all other information after the `%m` replacement character is also truncated. As such, we strongly recommend using the `%m` replacement at the end of the subject line only.

`%%` It represents a single `%` sign. As an example, you may have the subject line set to `Event from %s: "m"` and enabled legacy processing. If a message `This is a test` were received from `172.16.0.1`, the resulting email subject would read: `Event from 172.16.0.1: This is a test`

In non-legacy mode, the Property Replacer can be used. With it, you can include any property from the event message and also modify it. Please visit the Property Replacer documentation for details.

As an example, in non-legacy mode, you can set the subject line to `Mesg: '%msg:1:15%' From: %fromhost%`. If the message `This is a lengthy test message` were received from `172.16.0.1`, the resulting email subject would read: `Mesg: 'This is a lengt' From: 172.16.0.1`. Please note that the message is truncated because you only extracted the first 15 characters from the message text (position 1 to 15).

Subject

File Configuration field:

szSMTPSubject

Description:

Subject line to be used for outgoing emails and it is used for each message sent. It can contain replacement characters or "Event Properties" to customize it with event details. This is especially useful when sending email

to cellular phones or pagers, which often display only the subject line and not the actual message body. The subject line – after expansion of the any replacement sequences – can hold a maximum of 255 characters. Characters beyond this will be truncated. Please note that many email systems impose a more strict limit and truncation may occur before the 255-character limit. It is advisable to limit the subject line length to 80 characters or less.

The mail body will also include full event information, including the source system, facility, priority, and actual message text as well as any other information that came with this event. As there is no size limitation for message bodies, the body always contains the full message received (except otherwise configured – see below).

Please note that Insert Menu entry allows you to add replacement characters e.g. `%msg%` - you can send out the actual message of an event in the subject line.

There will be one email for each received message. Email delivery is meant for urgent notifications and actions (e. g. calling pagers and such). It is not meant to provide an email report.

Please note that The message content of the Message field can be configured. Event properties are described in the property replacer section.

Mail Priority

File Configuration field:

nMailPriority

- 0 = low
- 1 = Default
- 2 = High

Description:

Here you can adjust the priority with which the mail will be sent. You can choose between “low”, “normal”, and “high” priority. With this you can give your setup some complexity, being able to send some events as “important” and others with less importance.

Mail Message Format

File Configuration field:

szSMTPBody

Description:

This is the format of the message body. Properties from the event can be included by using the Property Replacer. Please note that the message body is only sent if “Include Message/Event in Email Body” is checked.

Output Encoding

File Configuration field:

nOutputEncoding

Description:

Determines the character encoding mode.

Use XML to Report

File Configuration field:

nUseXMLtoReport

Description:

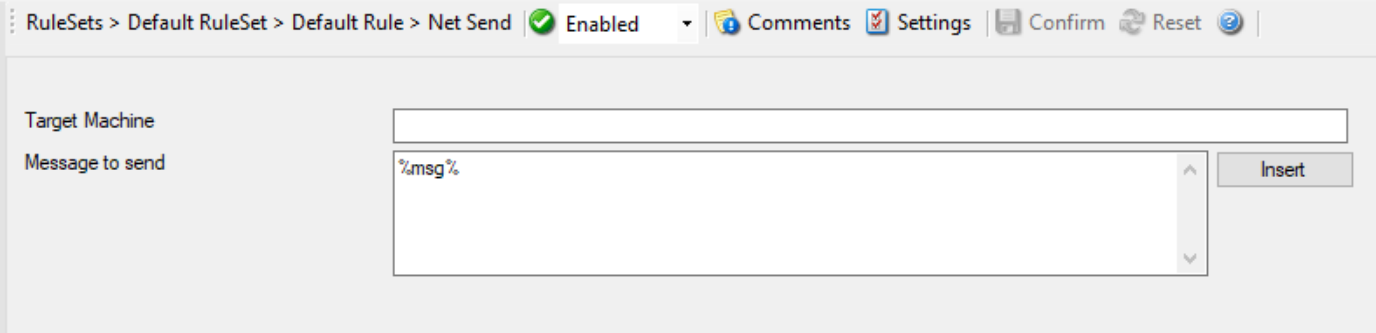
If checked, the received event will be included in XML format in the mail. If so, the event will include all information, like the original timestamp, the facility, priority etc. XML format is especially useful if the mail is sent to an automated system, which will then parse the message.

If unchecked, just the plain text message will be included in the mail. This format is more readable for a human reader.

Net Send

> **Warning**
>
> **Deprecated:** The Windows Messenger service (`net send` pop-up messages) is not available by default on modern Windows versions and may be blocked or unavailable in managed environments. As a result, delivery is often unreliable. Prefer modern alerting methods like the **Send Email** action or forwarding to syslog.

With the “Net Send” action, short alert messages can be sent via the Windows “net send” facility. These messages are delivered on a best-effort basis. If the recipient can be reached, they will pop up in a message box on the recipient’s machine. If the recipient cannot be reached, they will simply be discarded. No buffering takes place. Consequently, the rule engine does not check if the message can be delivered. It will never flag an action to be in error due to a reported delivery problem with “net send”.



- Action - Net Send*

Target Machine

File Configuration field:

szTarget

Description:

This is the Windows user name of the intended recipient, a NETBIOS machine name, or even an IP address (in the form of `10.1.1.1`). You can either use an IPv4, an IPv6 Address or a Hostname that resolves to an IPv4 or IPv6 Address.

Message to send

File Configuration field:

szMessage

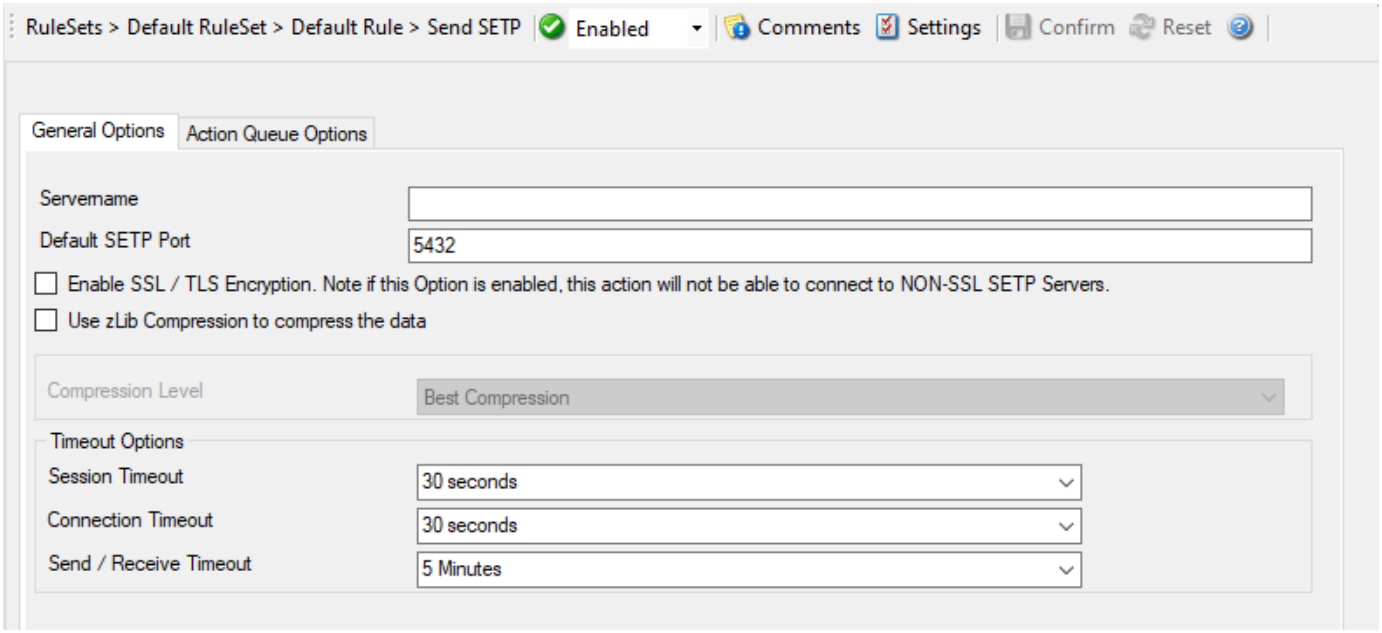
Description:

This is the message that is sent to the intended target.

Please note that the message content of the Message to send field can now be configured. event properties are described in the property replacer section.

Send SETP

With the “Send SETP” action, messages can be sent to a SETP server.



- Action - Send SETP General*

Servername

File Configuration field:

szServer

Description:

The product sends setp to the server/listener under this name. You can either use an IPv4, an IPv6 Address, or a Hostname that resolves to an IPv4 or IPv6 Address.

Default SETP Port

File Configuration field:

nMIAPSendPort

Description:

The Send setp sends outgoing requests on this port. The default value is 5432. Set the port to 0 to use the system-supplied default value (which defaults to 5432 if not modified by a system administrator).

Instead of the port number, a service name can be used. If so, that name is looked up via the socket service database functions. The lookup is for protocol TCP.

Please Note: The SETP port configured here must match the port configured at the listener side (for example, WinSyslog Enterprise edition). If they do not match, a Send SETP session cannot be initiated. The rule engine will log this to the Windows Event Log.

Enable SSL / TLS Encryption

File Configuration field:

nUseSSL

Description:

If this option is enabled then this action will be able to connect to SSL/TLS setp servers. Please make sure that you want this option to be enabled.

Use zLib Compression to compress the data

File Configuration field:

nZlibComp

Description:

It enables zLib compression support. Note that the SETP receiver must have zLib Compression support and enabled, otherwise it does not work.

Compression Level

File Configuration field:

nCompLevel

- 1 = Best Speed
- 3 = Low Compression
- 6 = Normal Compression
- 9 = Best Compression

Description:

Higher level results in better compression but slower performance.

Session Timeout

File Configuration field:

nTimeOutSession

Description:

The maximum time a session to a SETP server is to be kept open.

Connection Timeout

File Configuration field:

nConnectTimeOut

Description:

Maximum time a connection can take to connect or disconnect.

Send / Receive Timeout

File Configuration field:

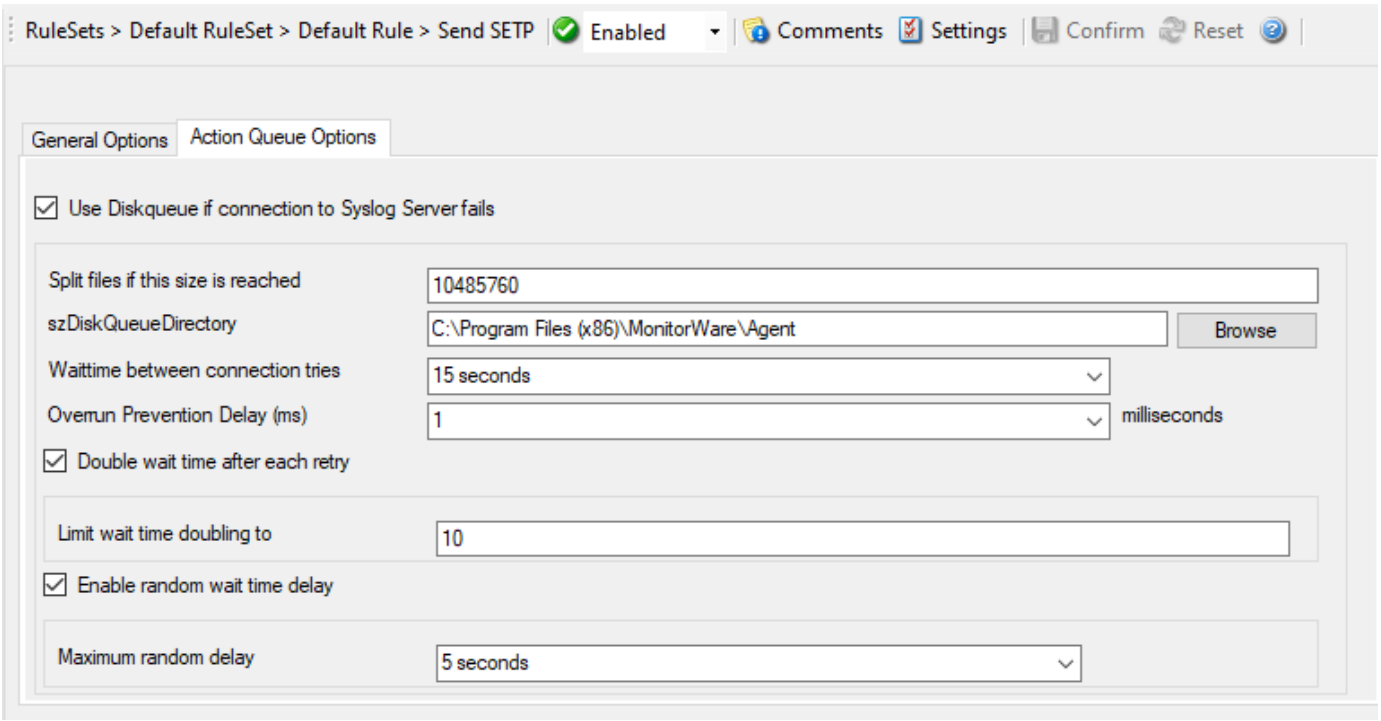
nSendRecvTimeOut

Description:

When sending or receiving data, this timeout applies.

Please note: If this option is enabled, this action is not be able to connect to NON-SSL SETP servers.

Action Queue Options



- Action - Send SETP Action Queue*

Use Diskqueue if connection to Syslog server fails

File Configuration field:

nUseDiscQueue

Description:

Enable diskqueuing syslog messages after unexpected connection loss.

Split files if this size is reached

File Configuration field:

nDiskQueueMaxFileSize

Description:

Files will be split until they reach the configured size in bytes. The maximum support file size is 10485760 bytes.

Diskqueue Directory

File Configuration field:

szDiskQueueDirectory

Description:

The directory where the queue files will be generated in. The queuefiles will be generated with a dynamic UUID bound to the action configuration.

Waittime between connection tries

File Configuration fields:

nDiskCacheWait

Description:

The minimum waittime until the Syslog Action retries to establish a connection to the Syslog server after failure.

Overrun Prevention Delay (ms)

File Configuration field:

nPreventOverrunDelay

Description:

When the Action is processing syslog cache files, an overrun prevention delay can be added to avoid flooding the target Syslog server.

Double wait time after each retry

File Configuration field:

bCacheWaittimeDoubling

Description:

If enabled, the configured waittime is doubled after each try.

Limit wait time doubling to

File Configuration field:

nCacheWaittimeDoublingTimes

Description:

How often the waittime is doubled after a failed connection try.

Enable random wait time delay

File Configuration field:

bCacheRandomDelay

Description:

If enabled, a some random time will be added into the waittime delay. When using many syslog senders, this can avoid that all senders start sending cached syslog data to the Syslog server at the same time.

Maximum random delay

File Configuration field:

nCacheRandomDelayTime

Description:

Maximum random delay time that will be added to the configured waittime if Enable random wait time delay is enabled.

Syslog Forwarding

Protocol Type

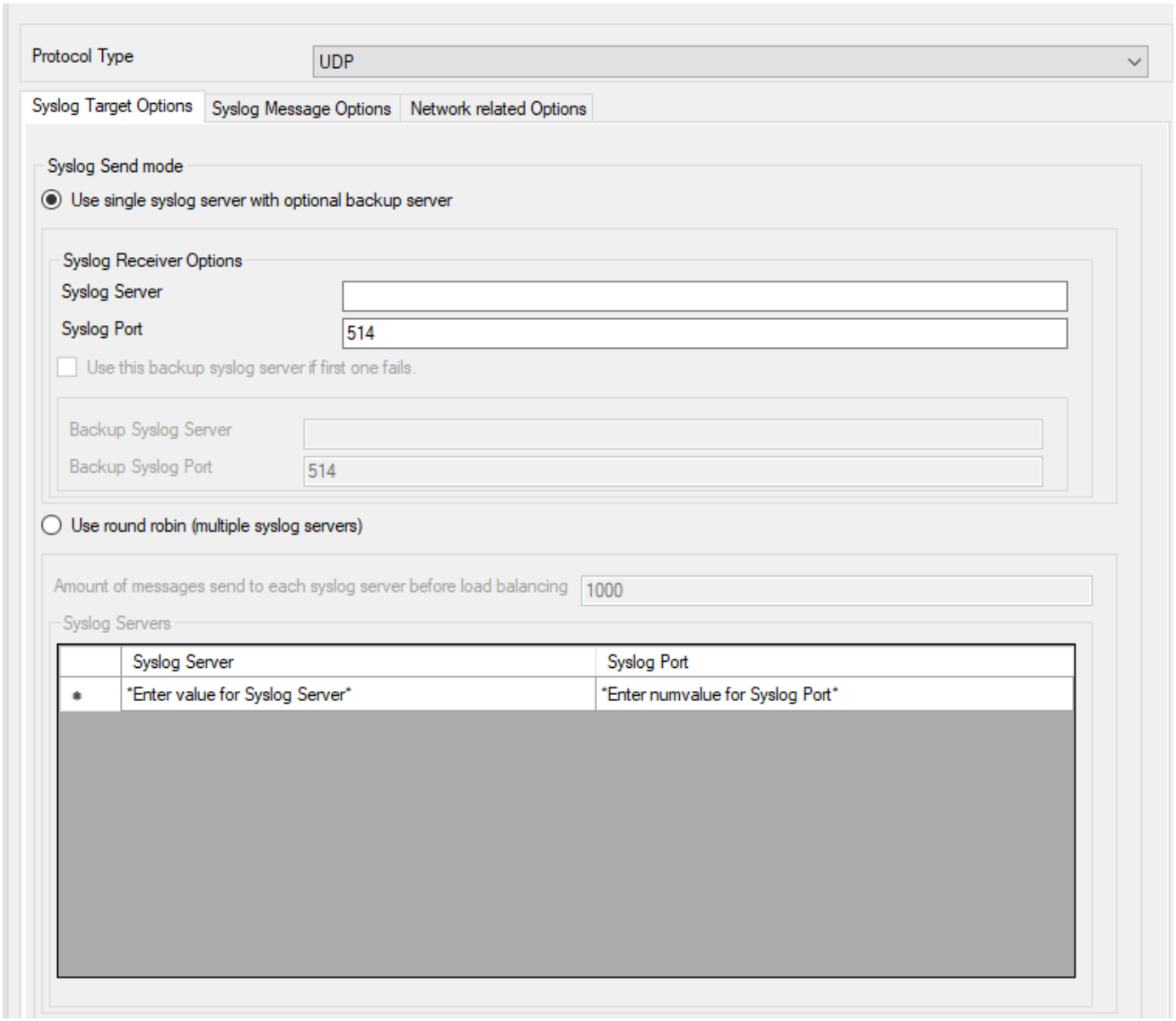
There are various ways to transmit syslog messages. In general,they can be sent via UDP, TCP, or RFC 3195 RAW. Typically, syslog messages are received via UDP protocol, which is the default. UDP is understood by almost all servers, but does not guarantee transport. In plain words, this means that syslog messages sent via UDP can get lost if there is a network error, the network is congested or a device (like a router or switch) is out of buffer space. Typically, UDP works quite well. However, it should not be used if the loss of a limited number of messages is not acceptable.

TCP and RFC 3195 based syslog messages offer much greater reliability. RFC 3195 is a special standardized transfer mode. However, it has not receive any importance in practice. Servers are hard to find. As one of the very few, Adiscon products support RFC 3195 also in the server implementations. Due to limited deployment, however, RFC 3195 is very little proven in practice. Thus we advise against using RFC 3195 mode if not strictly necessary (e.g. part of your requirement sheet).

TCP mode comes in three flavors. This stems back to the fact that transmission of syslog messages via plain TCP is not yet officially standardized (and it is doubtful if it ever will be). However, it is the most relevant and most widely implemented reliable transmission mode for syslog. It is a kind of unwritten industry standard. We support three different transmission modes offering the greatest compatibility with all existing implementations. The mode “TCP (one message per connection)” is a compatibility mode for Adiscon servers that are older than roughly June 2006. It may also be required for some other vendors. We recommend not to use this setting, except when needed. “TCP (persistent connection)” sends multiple messages over a single connection, which is held open for an extended period of time. This mode is compatible with almost all implementations and offers good performance. Some issues may occur if control characters are present in the syslog message, which typically should not happen. The mode “TCP (octet-count based framing)” implements algorithms of an IETF standard RFC 6587. It also uses a persistent connection. This mode is reliable and also deals with embedded control characters very well. This standard is now widely supported by modern syslog receivers and implementations.

As a rule of thumb, we recommend to use “TCP (octet-count based framing)” if you are dealing only with (newer) Adiscon products. Otherwise, “TCP (persistent connection)” is probably the best choice. If you select one of these options, you can also select a timeout. The connection is torn down if that timeout expires without a message being sent. We recommend to use the default of 30 minutes, which should be more than efficient. If an installation only occasionally sends messages, it could be useful to use a lower timeout value. This will free up connection slots on the server machine.

Syslog Target Options



- Action - Forward Syslog Target Options*

Syslog Send mode

File Configuration field:

nSendMode

Description

The Sendmode has been added since 2018 into all products supporting the forward syslog action. There are two options available.

Use single Syslog server with optional backup server This is the classic syslog send mode which uses a primary Syslog server and a secondary backup Syslog server if configured.

Use round robin (multiple syslog servers) This new method allows you to configure multiple targets that will be used one by one after a configured amount of messages has been sent to each target.

Syslog server (Syslog Send mode)

File Configuration field:

szSyslogSendServer

Description:

This is the name or IP address of the system to which Syslog messages should be sent to. You can either use an IPv4, an IPv6 Address, or a Hostname that resolves to an IPv4 or IPv6 Address.

Syslog Port (Syslog Send mode)

File Configuration field:

nSyslogSendPort

Description:

The remote port on the Syslog server to report to. If in doubt, please leave it at the default value of 514, which is typically the Syslog port. Different values are only required for special setups, for example in security sensitive areas. Set the port to 0 to use the system-supplied default value (which defaults to 514 on almost all systems).

Instead of the port number, a service name can be used. If so, that name is looked up via the socket service database functions.

Use this backup Syslog server if first one fails

File Configuration field:

nEnableBackupServer

Description:

The backup server is automatically used if the connection to the primary server fails. The primary server is automatically retried when the next Syslog session is opened. This option is only available when using TCP syslog.

Use round robin (multiple Syslog server)

Amount of messages send to each Syslog server before load balancing

File Configuration field:

nRoundRobinMsgCount

Description:

When using round robin mode, this is the amount of messages to be sent to each configured Syslog server.

Syslog server (Round robin mode)

File Configuration field:

szSyslogServer_[n]

Description:

This is the name or IP address of the system to which Syslog messages should be sent to. You can either use an IPv4, an IPv6 Address, or a Hostname that resolves to an IPv4 or IPv6 Address.

Syslog Port (Round robin mode)

File Configuration field:

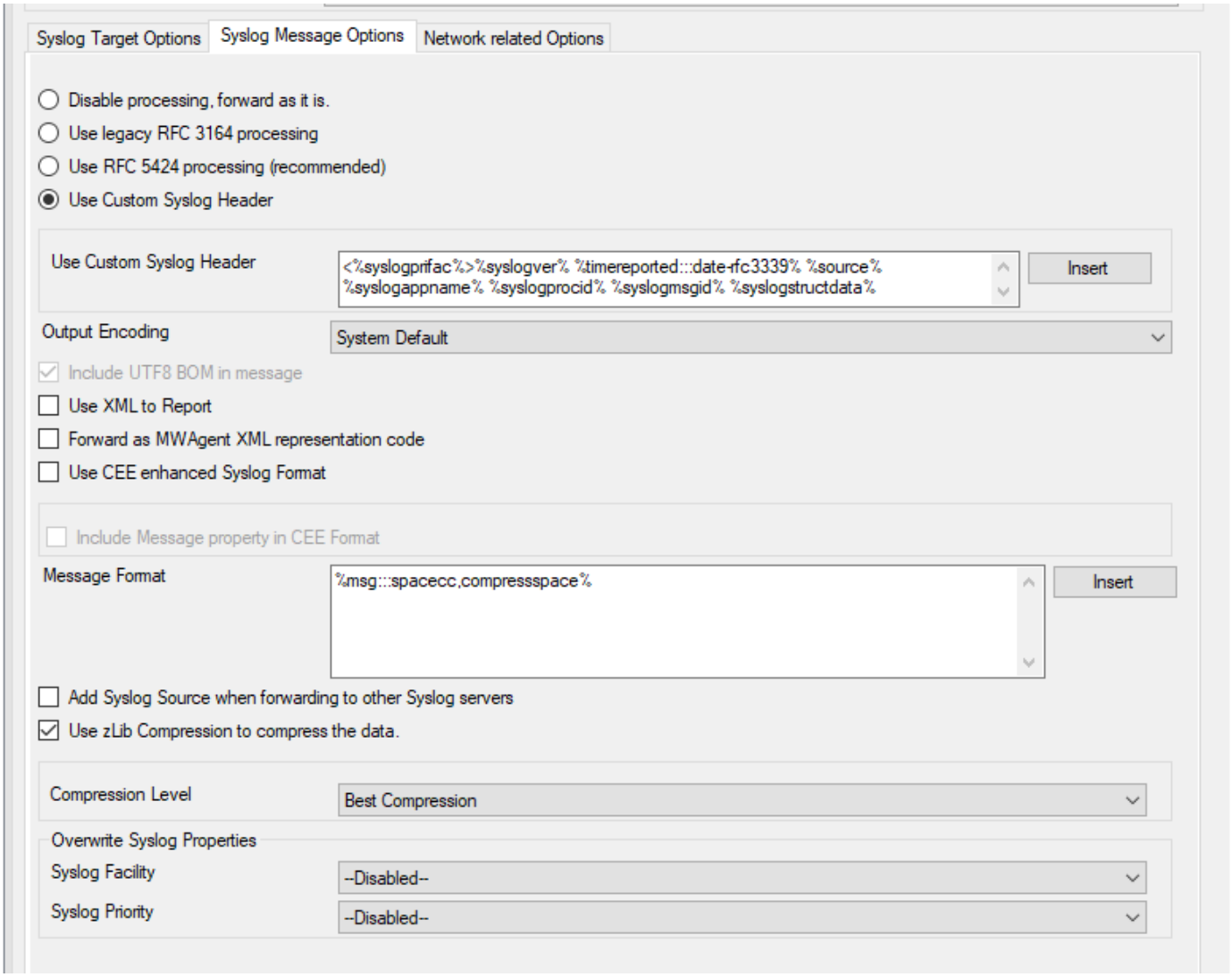
nSyslogPort_[n]

Description:

The remote port on the Syslog server to report to. If in doubt, please leave it at the default value of 514, which is typically the Syslog port. Different values are only required for special setups, for example in security sensitive areas. Set the port to 0 to use the system-supplied default value (which defaults to 514 on almost all systems).

Instead of the port number, a service name can be used. If so, that name is looked up via the socket service database functions.

Syslog Message Options



- Action - Forward Syslog - Message Options*

Syslog processing

File Configuration field:

bProcessDuringRelay

- 0 = Disable processing, forward as it is
- 1 = RFC3164 Header - Use legacy RFC 3164 processing
- 2 = RFC5424 Header - Use RFC 5424 processing (recommended)
- 3 = Custom Syslog Header

Description:

With this settings you can assign how your syslog messages will be processed.

For processing syslog you can choose out of four different options. You can use rfc3164 or RFC5424 (recommended) which is the current syslog standard, you are able to customize the syslog header or you do not process your syslog and forwards it as it is.

Use Custom Syslog Header

File Configuration field:

szCustomSyslogHeader

Description:

In this field you can specify the contents of your syslog header. This option is only available when you choose “Use Custom Syslog Header” in the Syslog Processing menu. The contents can be either a fixed message part which you can write into the field yourself or you use properties as dynamic content. By default the Header field is filled with the content of the RFC 5424 header.

Please note that the header content of the Header field can be configured. event properties are described in the property replacer section.

Output Encoding

File Configuration field:

nOutputEncoding

Description:

This setting is most important for Asian languages. A good rule is to leave it at “System Default” unless you definitely know you need a separate encoding. “System Default” works perfect in the far majority of cases, even on Asian (e.g. Japanese) Windows versions.

Include UTF8 BOM in message

File Configuration field:

nProtocolType

Description:

If enabled (default), the UTF8 BOM code will be prepended to the output message if you are using UTF8 Output encoding. If the syslog receiver cannot handle and remove the UTF8 BOM you can disabled this option.

Use XML to Report

File Configuration field:

bReportInXML

Description:

If this option is checked, the forwarded Syslog message is a complete XML-formatted information record. It includes additional information like timestamps or originating system in an easy to parse format.

The XML formatted message is especially useful if the receiving system is capable of parsing XML data. However, it might also be useful to a human reader as it includes additional information that cannot be transferred otherwise.

Forward as MonitorWare Agent XML Representation Code

File Configuration field:

nForwardIUT

Description:

MonitorWare supports a specific XML-Representation of the event. If it is checked, that XML representation is used. It provides additional information (like informationunit type, original source system, reception time & many more) but is harder to read by a human. At the same time, it is obviously easier to parse.

Use CEE enhanced Syslog Format

File Configuration field:

nReportInJSON

Description:

If enabled, the CEE enhanced Syslog format will be used. All useful properties will be included in a JSON Stream. The message itself can be included as well, see the “Include message property in CEE Format” option. Here is a sample how the format looks like for a security Eventlog message:

```
@cee: {"source": "machine.local", "nteventlogtype": "Security", "sourceproc": "Microsoft-Wi
ndows-Security-Auditing", "id": "4648", "categoryid": "12544", "category": "12544", "keyw
```

```
ordid": "0x8020000000000000", "user": "N\A", "SubjectUserSid": "S-1-5-11-222222222-33333333
3-4444444444-5555", "SubjectUserName": "User", "SubjectDomainName": "DOMAIN", "SubjectLogon
Id": "0x5efdd", "LogonGuid": "{00000000-0000-0000-0000-000000000000}", "TargetUserName": "A
dministrator", "TargetDomainName": " DOMAIN ", "TargetLogonGuid": "{00000000-0000-0000-0000
-000000000000}", "TargetServerName": "servername", "TargetInfo": " servername ", "ProcessId
": "0x76c", "ProcessName": "C:\Windows\System32\spoolsv.exe", "IpAddress": "-", "IpPort": "
-", "catname": "Logon", "keyword": "Audit Success", "level": "Information", }
```


Additionally to this format you can set: Include message property in CEE Format.

If enabled, the message itself will be included in the JSON Stream as property. Disable this option if you do not want the message itself in the CEE Format.

Please note you can also make Event ID part of the actual Syslog message while forwarding to a Syslog server. To do so, use XML output or a custom message format in the Forward Syslog action.

Include message property in CEE Format

Description

If enabled, the message itself will be included in the JSON Stream as property. Disable this option if you do not want the message itself in the CEE Format.

Please note you can also make Event ID part of the actual Syslog message while forwarding to a Syslog server. To do so, use XML output or a custom message format in the Forward Syslog action.

Message Format

File Configuration field:

szMessageFormat

Description:

The custom format lets you decide how the content of a syslog message looks like. You can use properties to insert content dynamically or have fixed messages that appear in every message. Event properties are described in the property replacer section.

Add Syslog Source when forwarding to other Syslog servers

File Configuration field:

nSyslogInsertSource

Description:

If this box is checked, information on the original originating system is prepended to the actual message text. This allows the recipient to track where the message originally came from.

Please note: This option is not compatible with RFC 3164. We recommend selecting it primarily when message forwarding to a WinSyslog Interactive Server is intended.

Use zLib Compression to compress the data

File Configuration field:

nUseCompression

Description:

With this option you can set the grade of compression for your syslog messages. For more information please read the note at the bottom of this page.

Compression Level

File Configuration field:

nCompressionLevel

- 1 = Best Speed

- 3 = Low Compression
- 6 = Normal Compression
- 9 = Best Compression (default)

Description:

With this option you can set the grade of compression for your syslog messages. For more information please read the note at the bottom of this page.

Note on Using Syslog Compression

Compressing syslog messages is a stable but rarely used feature. There is only a very limited set of receivers who are able to understand that format. Turning on compression can save valuable bandwidth in low-bandwidth environments. Depending on the message, the saving can be anything from no saving at all to about a reduction in half. The best savings ratios have been seen with Windows Event Log records in XML format. In this case, 50% or even a bit more can be saved. Very small messages do not compress at all. Typical syslog traffic in non-xml format is expected to compress around 10 to 25%.

Please note that compression over TCP connections requires a special transfer mode. This mode uses OpenSSL TLS Implementation 3.x for secure transmission. TLS compression is not implemented; instead, the system uses standard OpenSSL compression mechanisms.

Besides the fact that the mechanisms behind compression are experimental, the feature itself is solid.

Overwrite Syslog Properties

Syslog Facility

File Configuration field:

nSyslogFacility

Description:

When configured, will overwrite the Syslog Facility with the configured value.

Syslog Priority

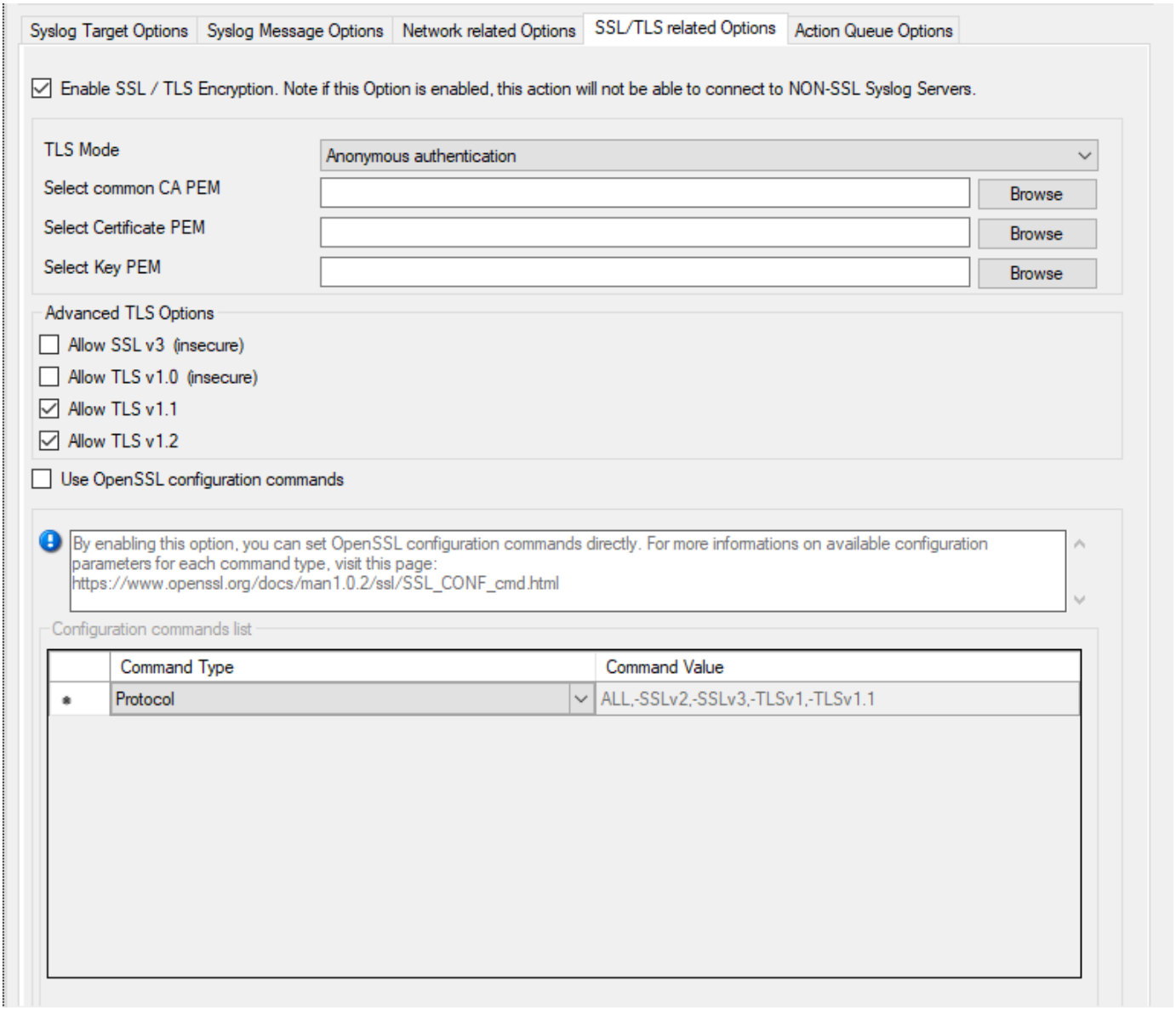
File Configuration field:

nSyslogPriority

Description:

When configured, will overwrite the Syslog Priority with the configured value.

SSL/TLS related Options



- Action - Forward Syslog SSL/TLS related Options*

Enable SSL / TLS Encryption

File Configuration field:

nUseSSL

Description:

If this option is enabled, the action will not be able to talk to a NON-SSL secured server. The method used for encryption is compatible to RFC5425 (Transport Layer Security (TLS) Transport Mapping for Syslog).

TLS Mode

File Configuration field:

nTLSMode

Description:

Anonymous Authentication

Default option. This means that a default certificate will be used.

Use Certificate

If this option is enable, you can specify your own certificate. For further authentication solutions, you will need to create your own certificates using OpenSSL Tools for example.

Select common CA PEM

File Configuration field:

szTLSCAFile

Description:

Select the certificate from the common Certificate Authority (CA). The syslog receiver should use the same CA.

Select Certificate PEM

File Configuration field:

szTLSCertFile

Description:

Select the client certificate (PEM Format).

Select Key PEM

File Configuration field:

szTLSKeyFile

Description:

Select the keyfile for the client certificate (PEM Format).

Allow SSL v3

File Configuration field:

nTLSAllowSSLv3

Description:

This option enables insecure protocol method SSLv3. We recommend NOT enabling this option as SSLv3 is considered broken.

Allow SSL v1.0

File Configuration field:

nTLSAllowTLS10

Description:

This option enables insecure protocol method TLSv1. We recommend NOT enabling this option as TLSv1 is considered broken.

Allow SSL v1.1

File Configuration field:

nTLSAllowTLS11

Description:

This option enables protocol method TLS1.1 which is enabled by default.

Allow SSL v1.2

File Configuration field:

nTLSAllowTLS12

Description:

This option enables protocol method TLS1.2 which is enabled by default.

Allow TLS v1.3

File Configuration field:

nTLSAllowTLS13

Description:

This option enables protocol method TLS1.3 which provides enhanced security and performance.

Use OpenSSL configuration commands

File Configuration field:

nTLSUseConfigurationCommands

Description:

By enabling this option, you can set OpenSSL configuration commands directly. For more information's on available configuration parameters for each command type, visit this page:

https://www.openssl.org/docs/man1.0.2/ssl/SSL_CONF_cmd.html

We allow to the set the following OpenSSL configuration commands in the configuration commands list:

- CipherString: Sets the allowed/disallowed used Ciphers. Setting this value will OVERWRITE the internal default ciphers.
- SignatureAlgorithms: This sets the supported signature algorithms for TLS v1.2.
- Curves: This sets the supported elliptic curves.
- Protocol: Sets the supported versions of the SSL or TLS protocol. This will OVERWRITE the Allow SSL options from above!
- Options: The value argument is a comma separated list of various flags to set.

When setting advanced configuration commands, we highly recommend to enable debug logging and review it after changes have been made. An error will be logged in the debug logfile if a configuration command cannot be processed successfully.

TCP related Options

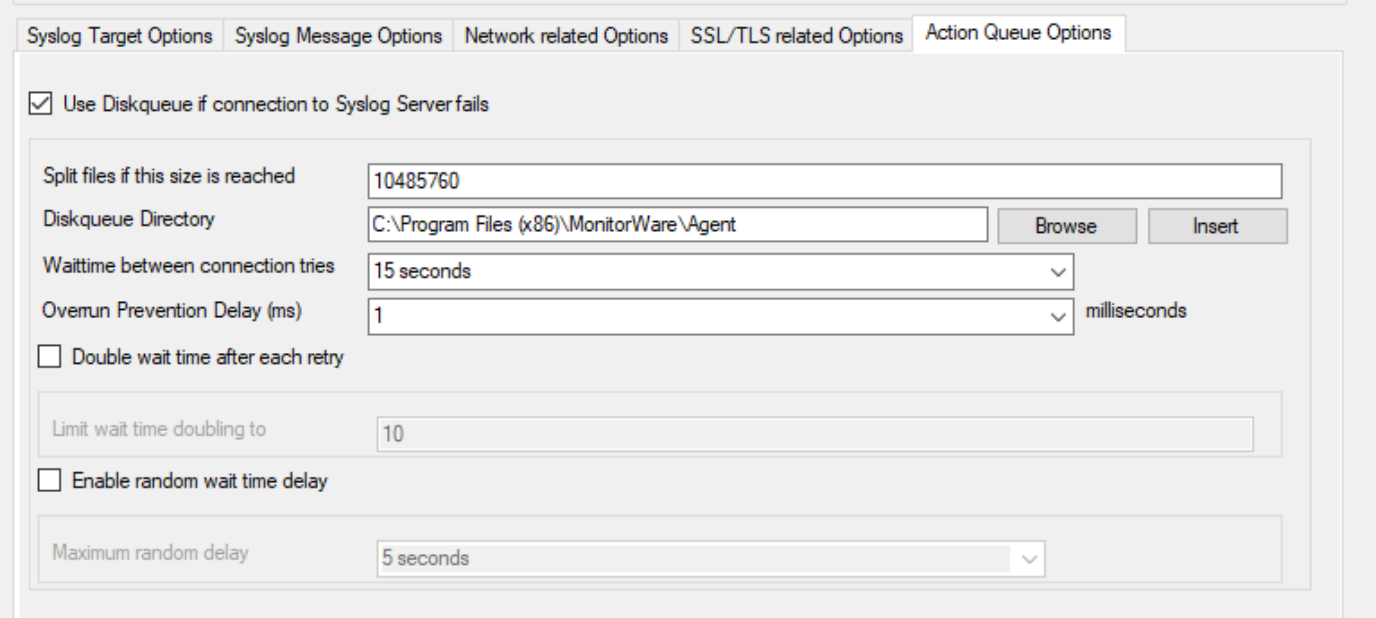
When using TCP-based syslog forwarding, you have the additional option to use the diskqueue. Whenever a connection to a remote Syslog server fails, the action starts caching the syslog messages into temporary files. The folder for these files can be configured. The filenames are generated using a unique GUID which is automatically generated for each Action, thus enabling you to use this feature in multiple Actions. Once the Syslog server becomes available again, the cached messages are being sent automatically. If you restart the Service while the Syslog Cache was active, it cannot be checked during service startup if the Syslog server is available now. Once the action is called again, the check is done and if the Syslog server is available, the messages are being sent. The size of this cache is only limited by the disk size. Files are split by 10MB by default, but this can also be configured. The maximum supported file size is 2GB.

Please Note: This option is not available for UDP or RFC 3195.

Session Timeout

File Configuration field: nTimeoutValue

Action Queue Options



- Action - Forward Syslog Action Queue*

Use Diskqueue if connection to Syslog server fails

File Configuration field:

nUseDiscQueue

Description:

Enable diskqueuing syslog messages after unexpected connection loss.

Split files if this size is reached

File Configuration field:

nDiskQueueMaxFileSize

Description:

Files will be split until they reach the configured size in bytes. The maximum support file size is 10485760 bytes.

Diskqueue Directory

File Configuration field:

szDiskQueueDirectory

Description:

The directory where the queue files will be generated in. The queuefiles will be generated with a dynamic UUID bound to the action configuration.

Waittime between connection tries

File Configuration fields:

nDiskCacheWait

Description:

The minimum waittime until the Syslog Action retries to establish a connection to the Syslog server after failure.

Overrun Prevention Delay (ms)

File Configuration field:

nPreventOverrunDelay

Description:

When the Action is processing syslog cache files, an overrun prevention delay can be added to avoid flooding the target Syslog server.

Double wait time after each retry

File Configuration field:

bCacheWaittimeDoubling

Description:

If enabled, the configured waittime is doubled after each try.

Limit wait time doubling to

File Configuration field:

nCacheWaittimeDoublingTimes

Description:

How often the waittime is doubled after a failed connection try.

Enable random wait time delay

File Configuration field:

bCacheRandomDelay

Description:

If enabled, a some random time will be added into the waittime delay. When using many syslog senders, this can avoid that all senders start sending cached syslog data to the Syslog server at the same time.

Maximum random delay

File Configuration field:

nCacheRandomDelayTime

Description:

Maximum random delay time that will be added to the configured waittime if Enable random wait time delay is enabled.

UDP related Options

Enable IP Spoofing for the UDP Protocol

File Configuration field:

nSpoofIPAddress

Description:

This option enables you to spoof the IP Address when sending Syslog messages over UDP. Some notes regarding the support of IP Spoofing. It is only supported the UDP Protocol and IPv4. IPv6 is not possible yet. Due system limitations introduced by Microsoft, IP Spoofing is only possible on Windows Server 2003, 2008, or higher. It is NOT possible in Windows XP, VISTA, 7, or higher. For more information see the Microsoft explanation. Also please note that most routers and gateways may drop network packages with spoofed IP Addresses, so it may only work in local networks.

Fixed IP or single property

File Configuration field:

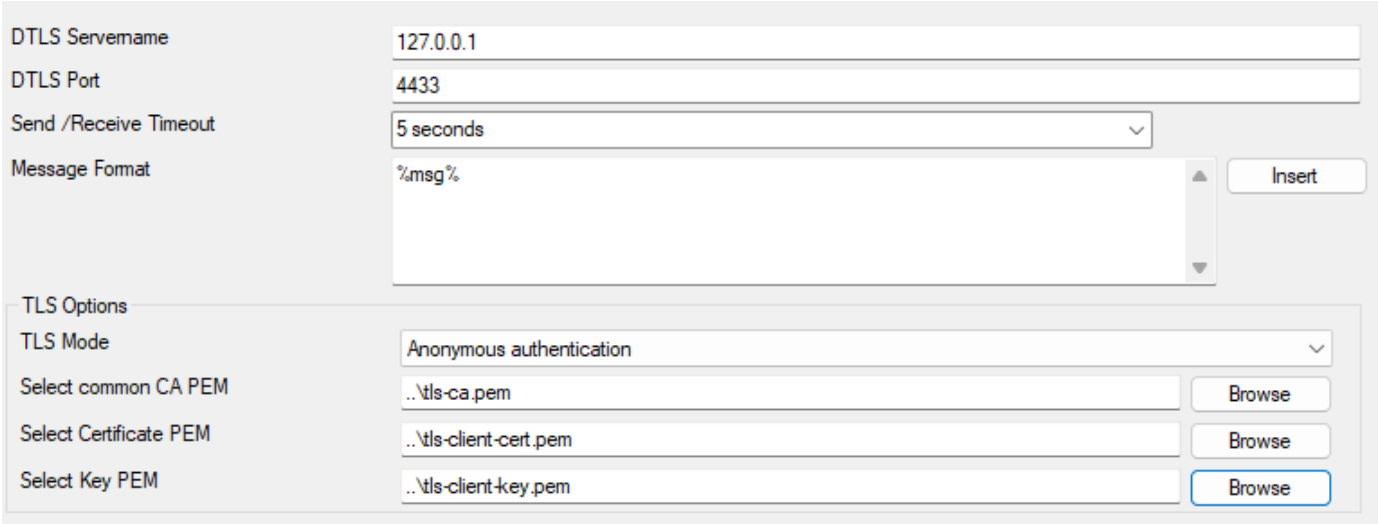
szSpoofedIPAddress

Description:

You can either use a static IP Address or a property. When using a property, the IP Address is tried to be resolved from the content of the property. For example by default the `%source%` property is used. If the name in this property cannot be resolved to an IP Address, the default local IP Address will be used.

Send DTLS

This action sends messages securely using the Datagram Transport Layer Security (DTLS) protocol. It ensures message confidentiality and integrity over an encrypted channel. The implementation uses OpenSSL to handle encryption and decryption, ensuring robust security and compatibility with industry standards. DTLS is suitable for applications requiring low latency and secure communication.



- Action - Send DTLS Configuration*

DTLS Servername

File Configuration field:

szDTLSServer

Description:

The hostname or IP address of the DTLS server to which messages should be sent. You can use an IPv4, IPv6 address, or a hostname resolving to one of these.

DTLS Port

File Configuration field:

nDTLSSendPort

Description:

The port number on the DTLS server where messages are sent. Typically, this port is 4433.

Send/Receive Timeout

File Configuration field:

nSendTimeOut

Description:

Specifies the time in seconds to wait for a response from the DTLS server before timing out. For instance, set this value to “5 seconds” for a 5-second timeout.

Message Format

File Configuration field:

szMessage

Description:

Defines the format of the message being sent. Use placeholders like “%msg%” to define the dynamic content of the message. Multi-line messages are supported.

TLS Options

TLS Mode

File Configuration field:

nTLSMode

Description:

Specifies the authentication method used. Options include “Anonymous authentication” or other modes requiring certificates.

Select common CA PEM

File Configuration field:

szTLSCAFile

Description:

Path to the CA certificate file (e.g., *tls-ca.pem*).

Select Certificate PEM

File Configuration field:

szTLSCertFile

Description:

Path to the client certificate file (e.g., *tls-client-cert.pem*).

Select Key PEM

File Configuration field:

szTLSKeyFile

Description:

Path to the private key file for the client (e.g., *tls-client-key.pem*).

Internal actions

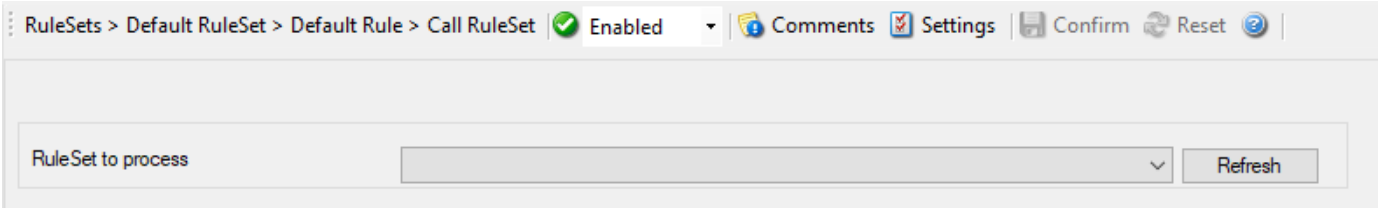
Call RuleSet

A Call RuleSet action simply calls another ruleset in some existing ruleset. When this action is encountered, the rule engine leaves the normal flow and goes to the called ruleset (which may contain many rules as well). It executes all the rules that have been defined in the called RuleSet. After the execution of all of them, it will return to its point from where it left the original flow. Let's take an example to clarify it a little further.

Let's say that Rule 1 has two actions - Action 1 and Action 2. The Action 1 of Rule 1 is an include (Call Ruleset) action. If the filter condition result of Rule 1 evaluates to true, it will execute the Action 1. Since Action 1 is the include action in this example, it will go to the included ruleset and will execute its filter condition. If that filter condition evaluates to true, it will execute all of its actions and will return to Action 2 of Rule 1 (of normal flow). If on the other hand, the filter condition of the included rule set evaluates to false, it will skip all of its actions and will come back to the Action 2 of Rule 1 (of normal flow).

Note: there is no limit on including the rules which means that a rule

that has been included in another rule may contain another rule in it which might contain another rule in it and so on.



- Action - Call RuleSet*

Ruleset to Call

File Configuration field:

szRuleSet

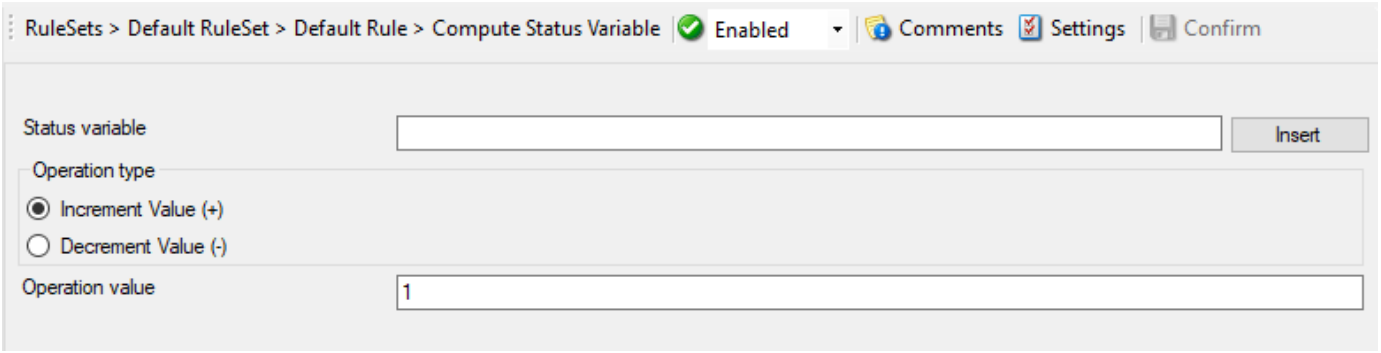
Description:

Select the Ruleset to be called.

Note: Call RuleSet stays disabled until you have more then "One" RuleSet!

Compute Status Variable

An internal action used to compute a status variable. This is needed for RuleSets which operate on a counter basis. This dialog controls the compute status variable options.



- Action - Compute Status Variable*

Status variable

File Configuration field:

szStatusVar

Description:

Name of the unique status variable.

Operation Type

Increment Value

File Configuration field:

nCalcType = 1

Description:

It increments the value by the operation value.

Decrement Value

File Configuration field:

nCalcType = 2

Description:

It decrements the value by the operation value.

Operation value

File Configuration field:

nChangeVal

Description:

The operation value that is to be used.

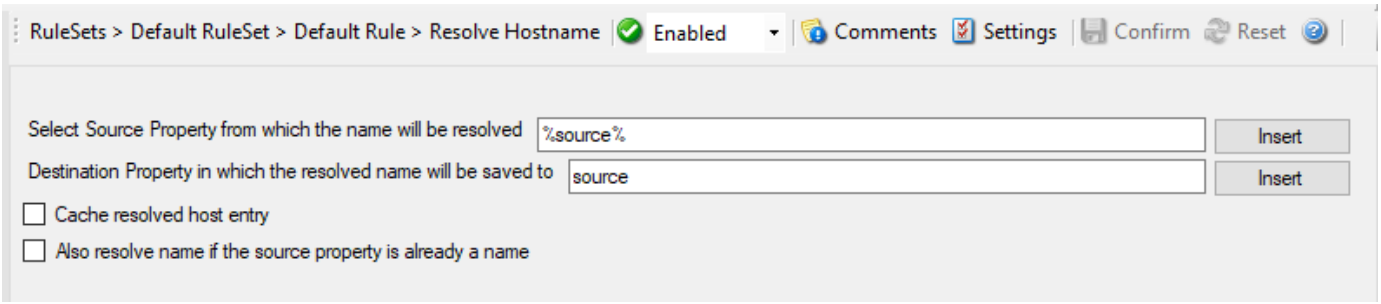
Discard

A Discard Action immediately destroys the current Information Unit and any action of any rule that has been defined after the Discard action execution. When this action has been selected then no dialog appears as nothing needs to be configured for this.

Use the Discard action when matching events should be ignored and no later actions in the same rule chain should run for them.

Resolve Hostname Action

Many Customers asked for resolve hostname options in different services. This feature has now been implemented as an action. An action can be used with every service, and it does not delay the work of a service.



- Action - Resolve Hostname*

Select Source Property from which the name will be resolved

File Configuration field:

szSourcePropertyName

Description:

Click on the Insert menu link on the right side of the textfield to customize the source property from which the name will be resolved.

Destination Property in which the resolved name will be saved to

File Configuration field:

szDestinationPropertyName

Description:

Same as above, please click on the Insert menu link on the right side of the textfield to customize the destination property in which the resolved name will be saved to.

Also resolve name if the source property is already a name

File Configuration field:

nResolveIfName

Description:

Activates the feature that the name will also be resolved if there is already a source property with that name.

Cache resolved host entry

File Configuration field:

nCacheNameEntry

Description:

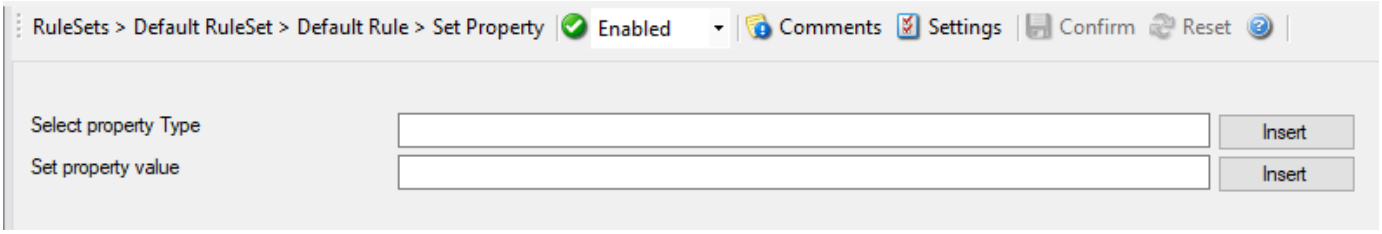
If activated this will, as it says, cache the resolved host entry.

Set Property

You can set every property and custom properties using this action.

This dialog controls the set property options. With the “Set Property” action, some properties of the incoming message can be modified. This is especially useful if an administrator would like to e.g. rename two equally named devices.

Please note: when you change or create a property, the value will be changed as soon as the set property action is carried out. It will not change before that happens and the old value is no longer available thereafter. That means all actions and filter conditions will use the new value after it is set. So, if you would like e.g. rename a system, make sure the set property actions are at the top of the rule base!



- Action - Set Property*

Select Property Type

File Configuration field:

szPropertyType

Description:

Select the property type to be changed. The list box contains all properties that can be changed. By default it is set to nothing.

Please note that the field content can be configured with event properties are described in the property replacer section.

Set Property Value

File Configuration field:

szPropertyValue

Description:

The value to be assigned to the property. Any valid property type value can be entered.

Please note that the field content can be configured with event properties are described in the property replacer section.

Difference from Set Status

Set Property changes data on the current message. The updated value is then used by later filters and actions for that same message only.

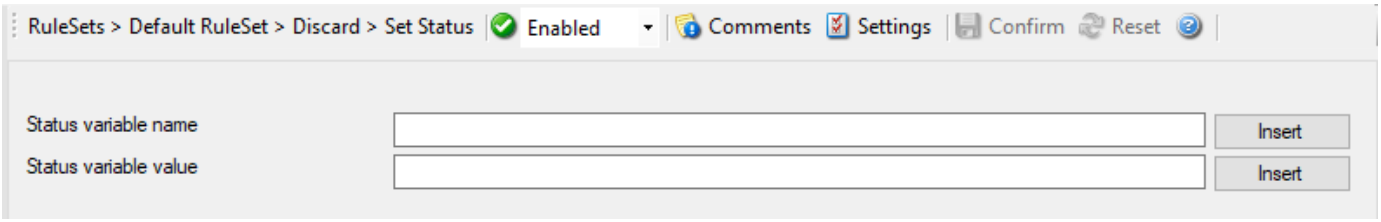
Use *Set Property* when you need to rewrite, normalize, or enrich message content. Use Set Status instead when you need a global value that persists across multiple messages.

Set Status

Each information unit have specific properties e.g. EventID, Priority, Facility etc. These properties have some values. Lets suppose that EventID has property value 01. Now, If you want to add “a new property of your own choice” in the existing set of properties then Set Status action allows you to accomplish this!

You can create a new property and assign any valid desired value to it e.g. we create a new property as CustomerID and set its value to 01. After you have created the property through this action, then you can define filters for them. There is an internal status list within the product which you can use for more complex filtering.

Please note: when you change a property, the value will be changed as soon as the set status action is carried out. It will not change before that happens and the old value is no longer available thereafter. That means all actions and filter conditions will use the new value after it is set. So if you would like e.g. rename a system, make sure the set status actions are at the top of the rule base!



- Action - Set Status*

Status Variable Name

File Configuration field:

szPropertyName

Description:

Enter the Property name. That name will from now on be used inside the rule base. More precisely, it will be used in the filter conditions and actions.

Please note that the field content can be configured with event properties are described in the property replacer section.

Status Variable Value

File Configuration field:

szPropertyValue

Description:

The value to be assigned to the property. Any valid property type value can be entered.

Please note that the field content can be configured with event properties are described in the property replacer section.

Difference from Set Property

Set Status and *Set Property* look similar, but they solve different problems:

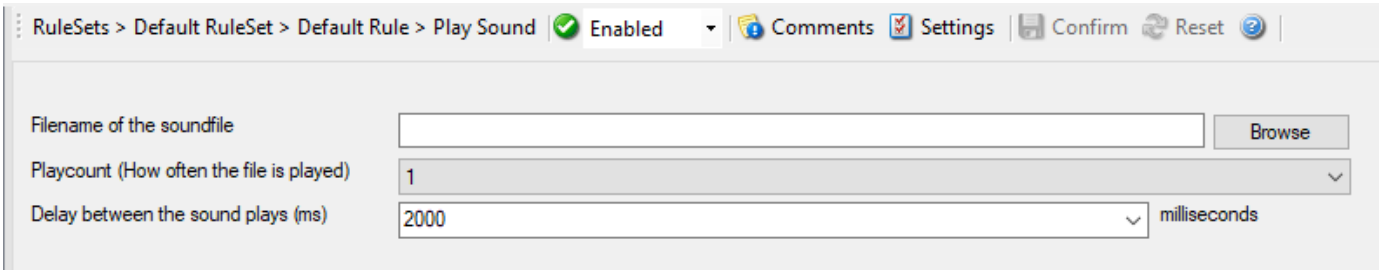
- A **property** belongs to the current message and exists only while that message is processed.
- A **status variable** is global and remains available across multiple messages until another action changes it.

Use *Set Status* when you need shared state across messages, for example a counter, workflow state, or a flag that later filters can evaluate.

Other actions

Play Sound

This action allows you to play a sound file. Since Windows VISTA/2008/7, Microsoft has disabled any interaction between a system service and the user desktop. This includes playing sounds as well. So if you want to use the Play Sound Action on any of this Windows Version, you will need to run the service in console mode (From command prompt with the -r option).



- Action - Play Sound*

Please note: if your machine has multiple sound cards installed, the **Play Sound** action uses the playback device that Windows exposes as the primary output for the service context.

If you need a different playback device, run the service under a user account whose Windows audio settings select the desired primary output. This is an advanced workaround and is usually of limited practical value on modern server systems.

Filename of the Soundfile

File Configuration field:

szFilename

Description:

Please enter the name of the sound file to play. **This must be a .WAV** file, other formats (like MP3) are not supported. While in theory it is possible that the sound file resides on a different machine, we highly recommend using files on the local machine only. Using remote files is officially not supported (but currently doable if you are prepared for some extra effort in getting this going).

If the file can either not be found or is not in a valid format, a system beep is emitted instead (this should - by API definition - be possible on any system).

Playcount

File Configuration field:

nCount

Description:

This specifies how many times the file is played. It can be re-played up to a hundred times.

Delay between Plays

File Configuration field:

nDelay

Description:

If multiple repeats are specified, this is the amount of time that is to be waited for between each individual play.

Start Program

With the “Start Program” action, an external program can be run. Any valid Windows executable can be run. This includes actual programs (EXE files), as well as scripts like batch files (.BAT), or VB scripts (.vbs).

Start process can, for example, be combined with the service monitor to restart failed services. Another example application is a script that deletes temporary files if the disk space monitor detects a low space condition.

- Action - Start Program*

Command to execute

File Configuration field:

szCommand

Description:

This is the path of actual program file to be executed. This can be the path of any valid executable file. A relative file name can be specified if it can be found via the operating system default search path.

Use legacy parameter processing

File Configuration field:

nUseLegacyProcessing

Description:

When enabled, old style parameter processing is used. Otherwise all properties can be used.

Parameters

File Configuration field:

szParameters

Description:

These parameters are passed to the program executed. They are passed as command line parameters. There is no specific format – it is up to the script to interpret them.

Parameters can contain replacement characters to customize it with event details. This allows passing event data to the script. The following replacement characters can be used:

- %d Date and time in local time
- %s IP address or name (depending on the “resolve hostnames” setting) of the source system that sent the message.
- %f Numeric facility code of the received message
- %p Numeric priority code of the received message
- %m The message itself
- %% Represents a single % sign.

In the example above, replacement characters are being used. If a message “This is a test” were received from “172.16.0.1”, the script would be started with 3 parameters:

Parameter 1 would be the string “e1” – it is assumed that this has some meaning to the script. Parameter 2 would be the IP address, 172.16.0.1. Parameter 3 would be “This is a test”. Please note that due to the two

quotes (“), the message is interpreted as a single parameters. If they were missing, it would typically be split into several ones, with parameter 3 being “This”, 4 being “is”, and so on. So these quotes are very important!

Sync Timeout

File Configuration field:

nSyncTimeOut

Description:

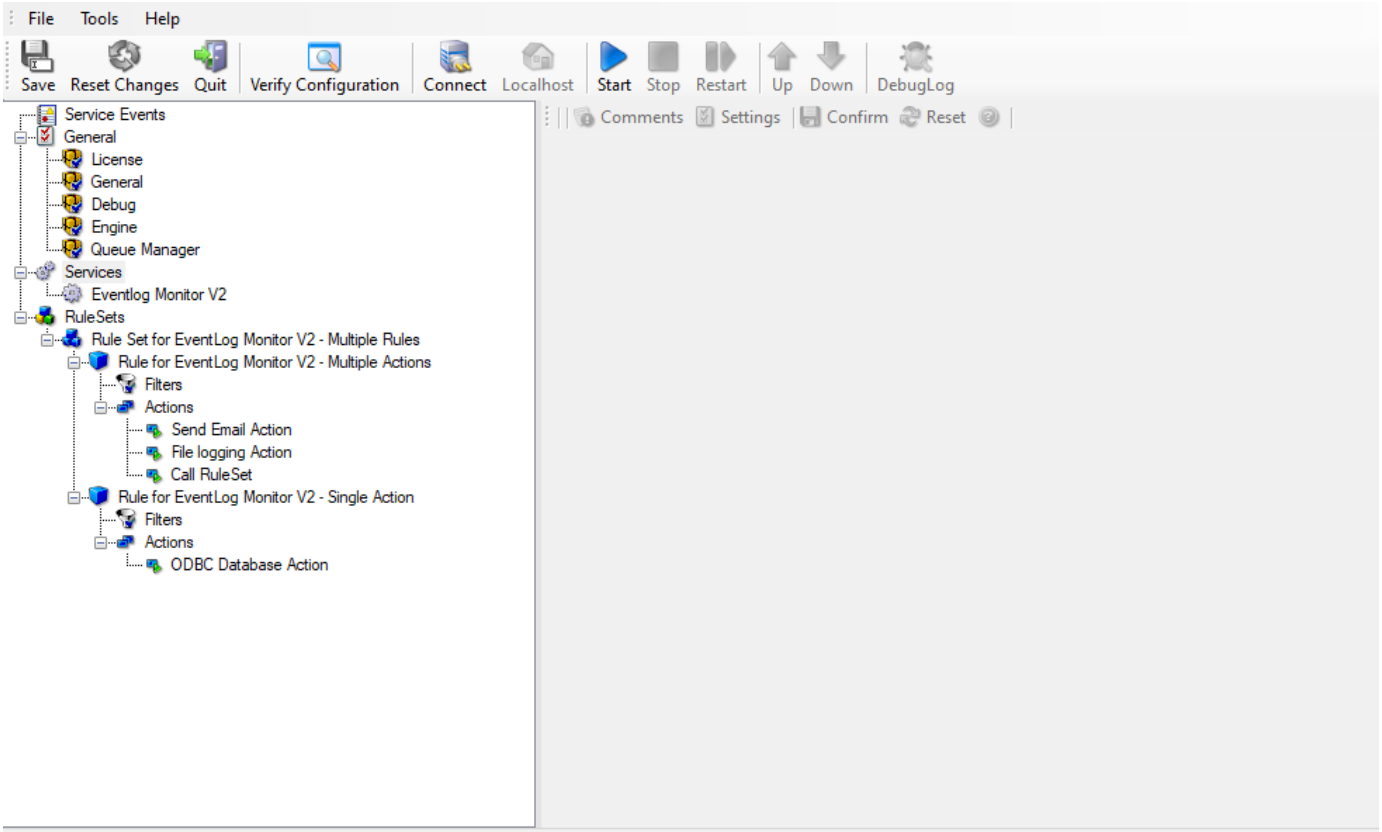
Time Out option is under Sync. Processing. When a program is executed, the service waits for it to finish before it carries on further actions. This is needed in order to ensure that all actions are carried out in the correct sequence.

The external program should only run for a limited amount of time. If it would block for some reason, the agent would be prevented from carrying out any further processing. As such, a timeout value must be specified. If the program still runs after the configured timeout, the rule engine cancels it, flags the action as unsuccessful, and then carries on with processing.

Important: Even though the timeout value can be as high as 30 seconds, we strongly recommend limiting the run time of external program to below 5 seconds. Otherwise, they could affect the overall performance too much. If the average run time is 5 seconds, the default timeout of 10 seconds ensures that the program can finish even when there is high system activity.

For performance reasons, we also strongly recommend to use the “Start Program” action only for rules that apply relatively seldom.

Multiple RuleSets - Rules - Actions



- Multiple RuleSets, Rules and Actions*

With EventReporter as many “RuleSets”, “Rules” and “Actions” as necessary can be defined.

You can create a separate “RuleSet” for each Service used, or just one “RuleSet” for all services. RuleSets can also be created to use them with the “callruleset action”.

RuleSets contain one or multiple rules. All actions and processing carried out is defined by the rules.

A rule consists of **`filter conditions`** and 1 to multiple actions. All actions that have to meet the same filter conditions can be combined in the same rule.

FAQ

Use this section for direct answers to common EventReporter questions. The entries below are grouped by user intent so you can scan by task instead of by product component.

Operation and troubleshooting

Why were dynamic log files sometimes not rotated in older EventReporter versions?

Older EventReporter versions could miss some scheduled rotations when dynamic filenames were combined with file-handle cleanup. That historical behavior was improved in later versions of the shared log rotation subsystem.

For the guarantees of the current subsystem, including graceful shutdown, restart recovery, reload behavior, and dynamic filename support, see **`../../shared/faq/log-rotation-guarantees`**.

Background

In older EventReporter versions, file-handle cleanup and scheduled rotation could interfere with each other when a dynamic log file had already become inactive before the scheduled rotation time arrived. Later versions improved the shared log rotation subsystem to handle this more reliably.

The Problem

Users may experience inconsistent log file rotation behavior where:

- Some log files rotate successfully every day as scheduled
- Some log files rotate only partially (not every day)
- Some log files never rotate at all

This typically occurs when log rotation is scheduled at specific times (e.g., at 0:00 every day) or when using dynamic filenames with property replacer.

Root Cause

The issue can be related to several factors:

- **File Handle Management:** When dynamic filenames are used, file handles are cached internally to avoid excessive file open/close operations. If files become inactive for extended periods, the cached handles may not be properly managed during rotation.
- **Timing Issues:** The rotation process may conflict with file access patterns, especially when multiple processes are accessing log files simultaneously.
- **Resource Constraints:** In high-volume logging environments, system resources may become constrained, affecting the rotation process.

Affected Versions

This issue affects EventReporter versions 18.5 to 19.1. Later versions include improvements to the log rotation logic that handle these scenarios more reliably.

Solutions

Recommended Solution: Upgrade EventReporter

The most effective solution is to upgrade to EventReporter version 19.1 or later, where the log rotation logic has been improved to handle these scenarios more reliably.

Alternative Solutions

If upgrading is not immediately possible, try these workarounds:

1. **Adjust File Handle Management:** * Review your file logging configuration * Ensure property replacer settings in filenames are optimized * Consider simplifying dynamic filename patterns if possible
2. **Modify Rotation Timing:** * Schedule rotations during periods of lower activity * Avoid scheduling rotations at exactly 00:00 when possible * Consider using size-based rotation instead of time-based rotation
3. **Resource Optimization:** * Monitor system resources during peak logging periods * Ensure adequate disk space is available * Consider load balancing if multiple instances are logging simultaneously

Important: Monitor your logging environment closely when implementing these workarounds, as they may affect overall system performance. .. _log-rotation-file-handle-cleanup-eventreporter:

Why were dynamic log files sometimes not rotated in older EventReporter versions?

Older EventReporter versions could miss some scheduled rotations when dynamic filenames were combined with file-handle cleanup. That historical behavior was improved in later versions of the shared log rotation subsystem.

For the guarantees of the current subsystem, including graceful shutdown, restart recovery, reload behavior, and dynamic filename support, see **`../../shared/faq/log-rotation-guarantees`**.

If you are still running one of the older affected builds, increasing the file handle cleanup timeout can reduce the chance of a missed scheduled rotation, but the recommended long-term solution is to upgrade to a current release.

Log Rotation Naming Convention Change in EventReporter 19.x

This article explains the change in rotated log file naming convention in EventReporter 19.x and later versions.

Question

Why are my rotated log files named differently after upgrading to EventReporter 19.x?

Answer

EventReporter 19.x and later versions use a new naming convention for rotated log files. Instead of placing sequence numbers before the file extension (e.g., `syslog1.csv`, `syslog2.csv`), the new format appends sequence numbers after the file extension (e.g., `syslog.csv.1`, `syslog.csv.2`).

This change is intentional and by design. The new format follows Unix/POSIX conventions and provides better compatibility with common log management tools and scripts.

What Changed

Old Format (EventReporter versions before 19.x):

- `syslog.csv` (active log file)
- `syslog1.csv` (first rotated file)
- `syslog2.csv` (second rotated file)
- `syslog3.csv` (third rotated file)

New Format (EventReporter 19.x and later):

- `syslog.csv` (active log file)
- `syslog.csv.1` (first rotated file)
- `syslog.csv.2` (second rotated file)
- `syslog.csv.3` (third rotated file)

Root Cause

This change was intentionally implemented as part of improvements to the log rotation subsystem. The new format provides several benefits:

1. **Better compatibility:** Follows Unix/POSIX conventions used by standard log rotation utilities
2. **Improved reliability:** Enhanced thread safety in the log rotation mechanism
3. **Tool compatibility:** Works better with common log management tools and scripts
4. **Industry standard:** Aligns with widely-adopted log rotation naming practices

Important: The filename format change cannot be reverted through configuration settings.

Impact on Existing Workflows

The naming convention change may affect:

- **Scripts that parse log filenames:** Scripts expecting the old format may fail to find rotated files
- **Monitoring tools:** Tools that reference specific filename patterns may need updates
- **Log management workflows:** Automated processes that depend on the old naming convention may break
- **Backup scripts:** File backup routines that filter by filename pattern may need adjustment

Solution

Updating Scripts and Tools

If you have scripts, monitoring tools, or applications that parse or reference rotated log files, update them to work with the new format:

1. **Modify file parsing logic:**

   Update patterns to handle `filename.ext.N` format instead of `filenameN.ext`

   Example (PowerShell):

   ```
   # Old pattern
   Get-ChildItem "syslog[0-9].csv"

   # New pattern
   Get-ChildItem "syslog.csv.[0-9]"
   ```

2. **Update hardcoded filename references:**

   Replace any hardcoded file paths in scripts to use the new naming convention
3. **Test compatibility:**

   Verify script functionality with the new naming convention in a test environment before deploying to production

Log Management Tool Compatibility

The new format is compatible with most modern log rotation utilities:

- **Standard logrotate:** Fully compatible with the new format
- **Third-party tools:** Most log management tools support POSIX-style naming
- **Custom solutions:** May require updates to filename matching patterns

Recommendation: Verify that your log management tools support the POSIX-style naming convention. Most modern tools do, but older or custom solutions may need configuration updates.

Migration Best Practices

When upgrading to EventReporter 19.x or later:

1. **Test in development:**
   - Deploy the new version in a development environment first
   - Run your log processing workflows and scripts
   - Verify all tools work correctly with the new naming format
   - Document any required changes
2. **Update automation:**
   - Modify scripts before deploying the new EventReporter version
   - Update monitoring tool configurations
   - Test all changes in the development environment
3. **Plan for transition:**

- Consider running both old and new versions during a transition period
- Update scripts to handle both naming conventions if needed during migration
- Document the change in deployment procedures

4. **Verify backup processes:**

   - Ensure backup scripts include files with the new naming pattern
   - Test backup restoration to verify rotated files are included
   - Update retention policies if they depend on filename patterns

5. **Update documentation:**

   - Document the filename format change in maintenance procedures
   - Update runbooks and operational guides
   - Communicate changes to all stakeholders

Common Questions

Can I configure EventReporter to use the old naming format?

No. The new naming format is built into the log rotation subsystem and cannot be changed through configuration. This ensures consistent behavior and maintains the reliability and thread safety improvements.

Will my existing rotated log files be renamed automatically?

No. Existing rotated files retain their original names. The new naming convention applies only to files rotated after upgrading to EventReporter 19.x or later.

What happens to old rotated files?

Old rotated files (using the previous naming convention) remain unchanged. They coexist with newly rotated files that use the new convention. You may want to rename old files manually if consistency is important for your workflows, or simply let them age out according to your retention policy.

Are there any performance implications?

No. The naming convention change does not affect performance. In fact, the underlying improvements to the log rotation subsystem provide better reliability and thread safety.

Additional Information

For more information about log rotation configuration, see the log file action documentation in the EventReporter manual.

If you need assistance updating scripts or tools to work with the new naming convention, contact Adiscon support at https://ticket.adiscon.com/

Why does log rotation fail when using ZIP compression in EventReporter?

This article explains why log rotation preparation can time out before the service detaches a file for background compression or move processing.

Problem

Log rotation preparation can fail when the “Maximum wait time for log rotation” setting is too short for the File Action to acquire the rotation mutex while the service is busy.

Symptoms

- Log files are compressed into ZIP format but remain in the live logging directory
- Rotation is skipped or deferred because the rotation preparation mutex could not be acquired in time
- Incomplete log rotation leaves compressed files in active directories
- Current day logs may be archived prematurely when using time-based rotation triggers

Root Cause

The “Maximum wait time for log rotation” setting in the EventReporter Configuration Client controls how long rotation preparation waits to acquire the File Action rotation mutex. Compression and move operations run afterward as detached background work. When this wait is too short, the file may not be detached for rotation under load.

Solution

Option 1: Increase Maximum Wait Time for Log Rotation

1. Open the EventReporter Configuration Client
2. Navigate to the rotation settings section
3. Locate the “Maximum wait time for log rotation” setting
4. Change the value from 15000 (15 seconds) to 60000 (60 seconds) or 120000 (120 seconds)
5. Save the configuration and restart the EventReporter service
6. Test log rotation during the next scheduled rotation period

Option 2: Switch to Size-Based Rotation

1. Change rotation trigger from time-based to size-based
2. Configure appropriate file size thresholds for rotation
3. Adjust rotation timing to avoid current-day log pre-archiving
4. Test rotation behavior with smaller log files first

Best Practices

- Set the rotation wait time to at least 60 seconds when the service is busy or storage is slow
- Consider increasing to 120 seconds for large log files or slow storage systems
- Use size-based rotation instead of time-based to prevent current-day log pre-archiving

Related Settings

- **Maximum wait time for log rotation**: Controls how long rotation preparation waits for the File Action rotation mutex (in milliseconds)
- **Rotation trigger**: Determines when rotation begins (time-based vs size-based)
- **Compression method**: Affects processing time (ZIP compression takes longer than other methods)

Verification

- Monitor log directories after rotation triggers to ensure files are properly moved
- Check that compressed files are not remaining in live logging directories
- Verify rotation completes within expected timeframes
- Confirm no rotation failures in EventReporter logs during test periods

Are EventReporter products affected by recent OpenSSL CVEs?

Question

Are EventReporter products affected by recent OpenSSL CVEs? Which OpenSSL version do the products use, and are the vulnerable components used?

Problem

Customers may see OpenSSL security advisories (e.g., multiple CVEs from OpenSSL 3.6, 3.5, 3.4, 3.3, 3.0, or 1.1.1/1.0.2 branches) and need to know:

- Whether EventReporter is affected by specific CVEs

- Which OpenSSL version is shipped with EventReporter
- Whether the vulnerable code paths or components are used

Symptoms

- Security or compliance teams request a formal assessment of OpenSSL CVEs for EventReporter
- Scans or reports may flag EventReporter due to bundled OpenSSL
- No observable runtime failure; this is a security/compliance assessment topic

Solution

EventReporter v19.x uses a specific OpenSSL version (e.g., 3.2.1). OpenSSL advisories list affected version ranges per CVE. Many CVEs affect only certain release branches (e.g., 3.6, 3.5, 3.4, 3.3, 3.0, 1.1.1, 1.0.2) and do **not** include all minor lines (e.g., 3.2.x).

If EventReporter ships OpenSSL from a branch that is not in the affected set for a given CVE, the product is not vulnerable to that CVE regardless of whether the vulnerable API exists in the code base.

Information:

- OpenSSL versions are embedded into the product statically without dependencies on system-installed versions
- The product uses its own bundled OpenSSL library, independent of any OpenSSL installation on the system
- This means system OpenSSL updates do not affect the product, and conversely, the product's OpenSSL does not affect system security

Important Notes:

- OpenSSL version information for your specific build can be obtained from Adiscon Support
- Adiscon monitors security advisories and provides updates as necessary
- For the most current information, consult the EventReporter release notes or contact Support

Notes

Troubleshooting the Start Program action in EventReporter

This article explains common issues with the Start Program action in EventReporter and provides solutions to resolve them.

Background

The Start Program action allows EventReporter to execute external programs, batch files, or scripts when specific Windows event log conditions are met. However, there are several common issues that can prevent this action from working correctly.

Common Issues and Solutions

Issue 1: Program not found or path problems

Symptoms: - The Start Program action appears to run but nothing happens - No error messages in the Windows Event Log - The external program works when run manually from command line

Root Cause: EventReporter may not be able to locate the executable file due to path issues or missing dependencies.

Solutions:

1. **Use absolute paths for all executables** - Instead of: `curl google.com > temp.txt` - Use: `C:\curl\curl-win\bin\curl.exe google.com > C:\temp\temp.txt`
2. **Verify executable location** - Check if the program exists in the specified path - Ensure all required DLL files are present - Test the command manually from Windows Command Prompt
3. **Check Windows PATH environment variable** - EventReporter may not have access to the same PATH as your user session - Use full paths instead of relying on PATH resolution

Issue 2: Permission problems

Symptoms: - No error messages in Event Log - Program works when run manually but not through EventReporter

Root Cause: EventReporter runs as a Windows service with different permissions than your user account.

Solutions:

1. **Store files in accessible locations** - Avoid system folders like `C:\Windows\System32` - Use generic folders like `C:\temp` or `C:\scripts` - Ensure EventReporter service has read/execute permissions
2. **Check file permissions** - Right-click on the executable file - Go to Properties > Security - Ensure “SYSTEM” and “SERVICE” accounts have execute permissions

Issue 3: Working directory problems

Symptoms: - Program runs but cannot find input/output files - Relative paths in scripts don’t work

Root Cause: The working directory when EventReporter executes the program may be different from expected.

Solutions:

1. **Use absolute paths for all file references** - Instead of: `> temp.txt` - Use: `> C:\temp\temp.txt`
2. **Set working directory in batch files** - Add `cd /d C:\your\working\directory` at the beginning of batch files

Issue 4: Event-specific parameter passing

Symptoms: - Program runs but doesn’t receive expected parameters - Event data is not passed correctly to the external program

Root Cause: EventReporter uses specific replacement characters to pass event data to external programs.

Solutions:

1. **Use correct replacement characters** - `%d` - Date and time in local time - `%s` - Source system IP address or name - `%f` - Numeric facility code - `%p` - Numeric priority code - `%m` - The event message itself - `%%` - Represents a single % sign
2. **Quote parameters properly** - Use quotes around parameters that contain spaces - Example: `"Event occurred: %m"` instead of `Event occurred: %m`

Troubleshooting Steps

1. **Check Windows Event Log** - Open Event Viewer (type “Event Viewer” in Windows search) - Navigate to Windows Logs > Application - Look for EventReporter-related error events
2. **Test with simple commands first** - Start with a basic batch file that creates a text file - Example: `echo Test > C:\temp\test.txt`
3. **Verify the command works manually** - Open Command Prompt as Administrator - Run the exact same command that EventReporter should execute - Ensure it works from the command line first
4. **Check EventReporter service account** - Verify which account EventReporter is running under - Ensure that account has necessary permissions
5. **Test event triggering** - Create a test event that should trigger your Start Program action - Verify the event is being detected by EventReporter - Check if the action is configured correctly

Example Working Configuration

Here’s an example of a properly configured Start Program action for EventReporter:

Command to execute: `C:\scripts\process-event.bat`

Parameters: `"%d" "%s" "%m"`

Batch file content (C:scriptsprocess-event.bat): ``batch @echo off
echo Event occurred at %1 from %2 >> C:\temp\event-log.txt
echo Message: %3 >> C:\temp\event-log.txt ``

Key points: - Full path to batch file - Quoted parameters to handle spaces in event messages - Absolute paths for output files - Proper use of replacement characters

Additional Tips

- **Timeout settings:** Keep external programs under 5 seconds runtime for best performance

- **Error handling:** Consider adding error checking to your batch files
- **Logging:** Add logging to your scripts to help troubleshoot issues
- **Testing:** Always test Start Program actions with actual Windows events
- **Event filtering:** Ensure your event filters are correctly configured to trigger the action

If you continue to experience issues after following these steps, please contact Adiscon support with: - EventReporter version - Windows version - Exact command being executed - Any error messages from Event Log - Results of manual command testing - Sample event that should trigger the action

Queue Buildup During SQL Server Table Cleanup Operations in EventReporter

This article explains queue buildup issues in EventReporter when performing regular cleanup operations on Microsoft SQL Server SystemEvents tables.

Question

Why does EventReporter's message queue build up when deleting old rows from the SQL Server SystemEvents table, even though the table is not explicitly locked?

Answer

When using Microsoft SQL Server as storage via OLEDB or ODBC Actions in EventReporter, performing regular cleanup operations (deleting old rows) on the SystemEvents table may cause EventReporter's message queue to build up even though the table is not explicitly locked. This can occur even with optimized batch delete processes that use primary key-based deletes.

Note: This issue applies specifically to EventReporter using Microsoft SQL Server as storage through OLEDB or ODBC Actions. The queue buildup occurs when cleanup operations (DELETE statements) run concurrently with EventReporter's INSERT operations to the same SQL Server database.

Symptoms

- Queue saturation incidents that correlate with cleanup schedule times
- Queue buildup during delete operations, even with batch delete processes
- Brief blocking or slowdowns during cleanup operations
- High memory consumption when queue holds large numbers of messages
- Processing rate barely keeping up with ingestion rate during cleanup

Root Cause

Even with optimized delete processes, several subtle mechanisms can cause contention between INSERT and DELETE operations:

1. **Page-Level Locks**: SQL Server may use page-level locks during deletes. If EventReporter's INSERT operations target the same pages being deleted, brief blocking can occur. With high-frequency inserts (200-400 messages per second or higher), even brief page-level contention can cause queue buildup.
2. **Index Maintenance Overhead**: The primary key index must be maintained during each delete batch. Even with efficient primary key-based deletes, index pages need to be updated, which can cause brief contention that slows INSERT operations.
3. **Transaction Log Activity**: Regular delete operations generate significant transaction log activity. If the transaction log is on the same disk as data files, I/O contention can occur during delete operations, temporarily slowing all database operations including inserts.
4. **Ghost Record Cleanup**: SQL Server marks deleted rows as "ghost records" initially, then cleans them up asynchronously. If ghost record cleanup coincides with high INSERT activity, page-level contention can occur.

These issues are more subtle than traditional locking problems and may not show up as explicit table locks, but can still cause queue buildup during delete operations.

Solution

Option 1: Enhance Delete Process with ROWLOCK Hint

The ROWLOCK hint forces SQL Server to keep locks at the row level instead of escalating to page-level locks, which means INSERT operations can proceed on other rows within the same pages even while deletes are running. Adding OPTION (MAXDOP 1) prevents parallel execution that could escalate locks.

Implementation:

```
DELETE TOP (5000) FROM SystemEvents WITH (ROWLOCK)
WHERE [Date] < DATEADD(day, -1, GETDATE())
OPTION (MAXDOP 1);
```

Benefits:

- Simple change that can reduce blocking
- No SQL Server edition changes required
- Works with Standard Edition
- Reduces lock escalation to page level

Considerations:

- Provides partial improvement, not complete elimination of contention
- Still requires index maintenance during deletes

Option 2: Separate Transaction Log Disk

Place the transaction log on a separate physical disk, separate from data files. This is a SQL Server best practice and eliminates I/O contention between log writes and data file operations.

Benefits:

- Eliminates I/O contention between log and data operations
- Works with any SQL Server edition
- Best practice for SQL Server configuration

Considerations:

- Requires separate physical disk or storage volume
- Addresses I/O contention only, not locking issues

Option 3: Table Partitioning (Enterprise Edition)

Partition the SystemEvents table by date (e.g., daily partitions). Inserts always go to today's partition while deletes target old partitions that no longer receive inserts, completely eliminating contention. When cleaning up, use partition switching to instantly move an entire partition to a staging table and drop it - this is a metadata-only operation that takes milliseconds instead of minutes.

Benefits:

- Completely eliminates contention between inserts and deletes
- Partition switching is a metadata-only operation (milliseconds)
- Inserts and deletes operate on different partitions

Considerations:

- Requires SQL Server Enterprise Edition
- Requires initial setup and ongoing partition management
- More complex implementation

Option 4: Delayed Durability (If Acceptable Risk)

This can improve insert performance by 20-40% by deferring transaction log writes. However, there's a risk of losing the last few seconds of inserts if the server crashes before the log is flushed.

Implementation:

```
ALTER DATABASE [DatabaseName] SET DELAYED_DURABILITY = ALLOWED;
```

Benefits:

- Significant performance improvement (20-40%)
- Works with Standard Edition
- Simple configuration change

Considerations:

- Risk of data loss if server crashes before log flush
- Addresses I/O performance only, not locking issues
- May not be acceptable for all environments

Best Practices

1. **Monitor Queue Saturation Correlation**: Check whether queue saturation incidents correlate with cleanup schedule times. If deletes run on a regular schedule and queue buildup occurs at predictable intervals, this strongly suggests the cleanup process is a contributing factor.
2. **Monitor SQL Server Blocking**: Use the following query to check for blocking during delete operations:

```
SELECT
r.session_id,
r.blocking_session_id,
r.wait_type,
r.wait_time,
SUBSTRING(t.text, 1, 200) AS QueryText
FROM sys.dm_exec_requests r
CROSS APPLY sys.dm_exec_sql_text(r.sql_handle) t
WHERE r.blocking_session_id > 0;
```

3. **Optimize Batch Delete Process**: Use primary key-based deletes with appropriate batch sizes (e.g., 5000 rows) to balance performance and lock duration. Using the primary key ensures efficient index seeks.
4. **Isolate Parsing Workload**: If using separate processes for parsing, use (NOLOCK) hints to completely isolate the parsing workload from EventReporter's insert operations.
5. **Verify Transaction Log Location**: Ensure transaction log is on separate physical disk from data files to eliminate I/O contention.
6. **Consider Action Queue Feature**: Use Action Queue feature at Database Action level (OLEDB or ODBC Actions) instead of increasing main queue limit excessively. This can help manage queue buildup during temporary database slowdowns.

Related Settings

- **Main Queue Limit**: The maximum number of messages that can be queued in EventReporter. Large limits (e.g., 2-4 million) can cause increased CPU overhead and memory consumption when queue is full.
- **Worker Threads**: Number of worker threads for parallel processing in EventReporter. Increasing worker threads (e.g., from 2 to 4) can improve parallel processing during normal operations.
- **Action Queue**: Feature at Database Action level (OLEDB or ODBC Actions) in EventReporter that provides additional buffering during temporary database slowdowns. Recommended over excessive main queue limits.
- **Database Connection Settings**: Connection timeout and retry settings for SQL Server connections via OLEDB or ODBC Actions in EventReporter.

Verification

To verify if cleanup process is contributing to queue issues in EventReporter:

1. **Check Timing Correlation**: Monitor whether queue saturation incidents occur at predictable intervals matching cleanup schedule.

2. **Monitor Blocking**: Run the blocking query during cleanup operations to identify any blocking sessions.
3. **Review SQL Server Wait Statistics**: Check for PAGEIOLATCH, LCK_M_* wait types during cleanup operations.
4. **Monitor Transaction Log Activity**: Check transaction log file growth and I/O during cleanup operations.
5. **Review Queue Metrics**: Monitor queue depth over time in EventReporter and correlate with cleanup schedule to identify patterns.

If queue buildup incidents correlate with cleanup schedule times, the recommendations described above can help address the contention issues.

Recommended Service Stop Order for EventReporter Maintenance

Question

What is the recommended order for stopping the EventReporter service during system maintenance or reboots?

Answer

When performing system maintenance, updates, or planned reboots on a system running EventReporter, follow a specific shutdown sequence to prevent data loss and ensure a clean shutdown. The EventReporter service should be stopped **after** any web server and **before** the database server.

Recommended Stop Order

1. **Stop IIS/Web Server** (if using a web-based log viewer)
2. **Stop EventReporter Service**
3. **Stop Database Server** (SQL Server, MySQL, etc.)

Rationale

This sequence ensures:

- **Web connections are closed first:** Prevents new user sessions from accessing the database while EventReporter is still writing.
- **EventReporter stops gracefully:** Allows EventReporter to complete any in-progress writes and flush its queues before the database becomes unavailable.
- **Database closes last:** Ensures all pending transactions from EventReporter are committed before the database shuts down.

Stop Commands

You can stop the EventReporter service using its internal service name `AdisconEvntSLog` in PowerShell or Command Prompt:

Command Prompt:

```
net stop w3svc
net stop "AdisconEvntSLog"
net stop MSSQLSERVER
```

PowerShell:

```
Stop-Service -Name "w3svc"
Stop-Service -Name "AdisconEvntSLog"
Stop-Service -Name "MSSQLSERVER"
```

Startup Order

When starting services after maintenance, reverse the order:

1. **Start Database Server**
2. **Start EventReporter Service**
3. **Start IIS/Web Server**

Command Prompt:

```
net start MSSQLSERVER
net start "AdisconEvntSLog"
net start w3svc
```

PowerShell:

```
Start-Service -Name "MSSQLSERVER"
Start-Service -Name "AdisconEvntSLog"
Start-Service -Name "w3svc"
```

Service Name Reference

When managing the EventReporter service from the command line, use the internal service name:

- **Internal Service Name:** `AdisconEvntSLog`
- **Display Name:** EventReporter Service

The internal service name remains consistent across installations and should be used in automation scripts for reliability.

Verifying Service Status

```
Get-Service -Name "AdisconEvntSLog"
```

```
sc query "AdisconEvntSLog"
```

Best Practices

- **Plan maintenance windows:** Schedule downtime during low-traffic periods to minimize event log message loss.
- **Backup database:** Perform a database backup before shutting down services.
- **Verify connections:** After restart, verify that the EventReporter service started correctly and is writing to the database.
- **Check logs:** Review the Windows Event Viewer for any EventReporter service errors after restart.
- **Use internal service names:** Always use the internal service name `AdisconEvntSLog` in scripts for reliability.

Troubleshooting

If the EventReporter service does not stop gracefully:

- Check for stuck processes in Task Manager.
- Review the Windows Event Viewer for service errors.
- Verify that database connections are properly closed.
- Check service dependencies: `sc qc "AdisconEvntSLog"`
- As a last resort, use force stop: `net stop "AdisconEvntSLog" /y`

Running EventReporter on a Windows Cluster Server

Question

Can EventReporter run on a Windows Cluster Server, and are there any particular issues to be aware of?

Answer

Yes. EventReporter runs on a Windows Cluster node without problems. However, EventReporter does not include built-in cluster failover support. If a node fails, you must start the EventReporter service on another node manually or through a cluster script. The steps below explain how to prepare the cluster for this scenario.

Step 1: Set the Service Startup Type

On every cluster node **except** the primary node, set the EventReporter service startup type to **Manual**:

1. Open the Windows Service Manager (Start > Control Panel > Administrative Tools > Services).
2. Locate the service named **AdisconEvntSLog**.
3. Right-click the service and select **Properties**.
4. Set **Startup type** to **Manual**.
5. Click **Apply**, then **OK**.

On the primary node, leave the startup type set to **Automatic** so that EventReporter starts automatically after a reboot.

Step 2: Mirror the Configuration Between Nodes

EventReporter stores its configuration in the Windows registry. To replicate a working configuration from one node to another, export it as a registry file and import it on each secondary node:

1. Open the EventReporter Configuration Client on the primary node.
2. Go to the **Computer** menu.
3. Select **Export Settings to Registry File**.
    - Choose the standard registry format (do **not** select a binary format).
    - Select the correct architecture (Win32 or x64) for your system.
4. Save the `.reg` file to a network share or removable media.
5. On each secondary node, double-click the `.reg` file to import the configuration.

After importing, the secondary node has the same configuration as the primary node. When a failover is needed, start the EventReporter service on the secondary node using the Services Manager or from the command line:

```
net start "AdisconEvntSLog"
```

Best Practices

- **Keep configurations in sync.** After every configuration change on the primary node, re-export and re-import the registry file on all secondary nodes.
- **Test failover regularly.** Verify that the EventReporter service starts correctly and processes events on each secondary node.
- **Use automation.** Consider a cluster resource script or a scheduled task that starts the EventReporter service on a secondary node when the primary node becomes unavailable.
- **Verify firewall rules.** Ensure that the necessary network ports are open on all cluster nodes so that event forwarding continues to work after failover.

How to Export EventReporter Settings for a Support Call

Question

How do I export the EventReporter configuration so I can send it to Adiscon support?

Answer

Open the EventReporter Configuration Client, export the settings as a text-based registry file, compress the exported file, and submit it through the Support Portal.

Details

Do **not** use the binary registry export format. Use the text-based export so support can inspect the configuration.

Action path

1. Open the EventReporter Configuration Client.
2. Go to **Computer -> Export Settings to Registry File**.
3. Save the file to a known location.
4. Compress the exported `.reg` file.
5. Send the ZIP archive through the support portal.

Related information

- Tutorial: Export the Configuration and Create a Debug Log

Why do log files remain locked when multiple rules write to the same file?

This article explains why WinSyslog, EventReporter, and MonitorWare Agent may continue to hold file handles to log files even after the configured timeout period, preventing external processes from accessing or archiving those files.

Applies To

- WinSyslog
- EventReporter
- MonitorWare Agent

Problem

The service continues to hold file handles to log files even after the configured timeout period has elapsed. This prevents external processes such as batch scripts or archiving tools from accessing, moving, or archiving the log files.

Symptoms

- Log files remain locked after the expected release time (e.g., after 2:00 AM daily when “Create unique filenames” is enabled)
- External batch scripts fail to archive log files because files are still in use by the service
- Files are not released even hours after the timeout period has passed
- CleanFileHandlesTimeout setting appears to be ignored or ineffective
- Error messages indicating files are in use when attempting to access them

Root Cause

The issue occurs when multiple file actions use identical filename templates that can reference the same physical file. This creates a conflict in file handle management:

1. Each rule’s file action creates its own independent file handle tracking mechanism
2. Multiple actions hold references to the same physical file simultaneously
3. The file cannot be released until ALL actions release their handles
4. Even though CleanFileHandlesTimeout is set correctly (e.g., 7200 seconds / 2 hours), the file remains locked because different actions have different timer states and may not all reach the timeout simultaneously

Problematic Configuration Pattern

Multiple rules using identical filename templates:

- File Path: `log-directory\%timegenerated%`
- File Base Name: `%source%-%timegenerated%.log`

When multiple rules match messages from the same source device, they all write to the same physical file (e.g., `device-ip-20251012.log`). This causes handle conflicts where each action maintains its own handle to the same file.

Example of the Issue

- Rule 1: Filter matches source “device-hostname” OR source “192.168.1.100” - File Action writes to `device-ip-20251012.log`
- Rule 2: Filter matches source “device-hostname” OR source “192.168.1.100” - File Action writes to `device-ip-20251012.log` (SAME FILE!)
- Rule 3-N: Same pattern with different filters but identical file action configuration

Solution

Option 1: Consolidate Rules with Identical File Actions (Recommended)

Instead of having multiple separate rules with identical file action configurations, consolidate them into ONE rule with all filter conditions combined. This ensures only one file action object manages each file.

Steps:

1. Create a new rule (e.g., “All Devices Combined”)
2. Use Copy & Paste functions in the configuration client to combine filters from all existing rules:
    - For each existing rule:
        - Navigate to the rule
        - Go to Filters tab
        - Find the filter group for that rule
        - RIGHT-CLICK on the filter block (usually an AND or OR block)
        - Select “Copy”
    - Navigate to your new “All Devices Combined” rule
    - Go to Filters tab
    - RIGHT-CLICK on the top-level OR filter
    - Select “Paste” (or “Paste as child”)
    - The entire filter block is now copied into your new rule
    - Repeat this process for all existing rules
3. Configure the file action once in the new combined rule
4. Delete the other rules so that only a single File Action remains

Result: Only ONE file action object manages each file, eliminating handle conflicts completely.

Option 2: Use Unique File Paths for Each Rule

If you need to keep separate rules for organizational purposes, ensure each rule writes to a unique file location:

- Organize logs into separate subdirectories by device type, rule name, or other categorization
- Use different filename patterns for each rule
- Ensure no two rules can write to the same physical file under any circumstances

Example Configuration:

- Rule 1: File Path: `log-directory\devices\type1\%timegenerated%`
- Rule 2: File Path: `log-directory\devices\type2\%timegenerated%`
- Rule 3: File Path: `log-directory\devices\type3\%timegenerated%`

Each rule writes to a completely separate directory structure, preventing any possibility of file handle conflicts.

Best Practices

- Each rule should write to a unique file location to avoid handle conflicts
- Consolidate rules with identical file action configurations into a single rule when possible

- Use the Copy & Paste functions in the configuration client to efficiently combine filter conditions from multiple rules
- When using “Create unique filenames” for daily rotation, ensure file paths are unique per rule
- Test file release behavior after configuration changes to verify files are released as expected
- Monitor file handles using Windows Resource Monitor or Process Explorer to verify proper release
- Verify that external batch scripts or archiving processes can access files after the timeout period

Verification

After implementing the solution:

1. Monitor file handles using Windows Resource Monitor or Process Explorer to confirm files are released after the CleanFileHandlesTimeout period
2. Verify configuration using the configuration client’s verification feature - it should report 0 errors related to duplicate filenames

If files are still not released after the timeout period, check for:

- Other file actions or rules that may be holding references to the same files
- Additional processes that may be accessing the files
- Configuration errors that may have been missed during the consolidation process

Is MariaDB supported by the ODBC action?

This article explains MariaDB support in ODBC database actions.

Question

Is MariaDB supported by the ODBC action?

Answer

Yes, MariaDB is fully supported by the ODBC action and can be used as a direct replacement for MySQL.

Background

MariaDB is a free and open-source alternative to MySQL. It is a fork of MySQL, initiated by the original MySQL developers after Oracle acquired Sun Microsystems (the former owner of MySQL). MariaDB was designed to be binary-compatible with MySQL, which generally makes switching from MySQL to MariaDB very easy.

Key characteristics of MariaDB:

- **Open Source:** MariaDB is consistently Open Source under a license that guarantees free use and further development
- **Binary Compatibility:** Designed to be binary-compatible with MySQL, making migration straightforward
- **Independent Development:** Continuous, independent development separate from MySQL
- **Performance:** Often preferred as an alternative due to sometimes better performance characteristics

Configuration

To use MariaDB with the ODBC action:

1. **Install MariaDB ODBC Driver:** - Download and install the MariaDB Connector/ODBC driver from the official MariaDB website - Ensure you install the correct version (32-bit or 64-bit) to match your Adiscon product installation
2. **Configure System DSN:** - Open the ODBC Data Source Administrator (use the 32-bit version if your product runs in 32-bit mode) - Create a new System DSN - Select the MariaDB ODBC driver - Configure the connection settings (server, database, credentials)
3. **Configure Database Action:** - In your Adiscon product configuration, select the ODBC Database action - Choose the MariaDB System DSN you created - Test the connection using the “Verify Database” button - Create the database tables if needed using the “Create Database” button

Note: The configuration process is identical to configuring MySQL, as MariaDB uses MySQL-compatible drivers and protocols.

Modern Deployment Recommendations

For current MariaDB deployments, we recommend the following:

1. **Use a current MariaDB server and connector** - Use currently supported MariaDB server releases - Use a current MariaDB Connector/ODBC package from the official source
2. **Use a dedicated database account** - Create a dedicated user for the Adiscon product - Grant only the required privileges on the target database/schema
3. **Enable secure transport for remote database connections** - Use TLS between the Adiscon host and MariaDB server when traffic crosses networks - Configure certificate settings in the DSN/driver according to your security policy
4. **Use UTF-8 consistently** - Prefer UTF-8/`utf8mb4` settings for server, database, and connector - This prevents character conversion issues in international log messages
5. **Validate end-to-end before production rollout** - Use “Verify Database” in the ODBC action - Insert sample messages and verify they are written and readable as expected

Common Modern Troubleshooting Checks

If connection tests fail, verify:

- Driver architecture matches the product runtime (32-bit vs 64-bit)
- Host, port, database name, and credentials in the DSN are correct
- MariaDB user authentication method is supported by the installed connector
- TLS requirements (if enabled) match server and connector configuration
- Firewall rules allow database traffic

Additional Information

For more information about database actions, see the ODBC Database Options documentation in your product’s manual.

For MariaDB-specific information, visit the official MariaDB website.

Deployment and administration

How Do I Perform a Repeatable Deployment of EventReporter?

Question

How can I deploy EventReporter consistently to multiple systems?

Answer

Use one system as the reference build, export its configuration, and automate distribution of the installer or service files plus the exported configuration through your normal software deployment tooling.

Details

A repeatable deployment is appropriate when more than one system should use the same or nearly the same EventReporter setup. The exact deployment mechanism depends on your environment, but the overall pattern stays the same:

- create and test one reference configuration
- export that configuration
- distribute the software and configuration together
- adjust only host-specific values where needed

This is a deployment method, not a size category. It is worthwhile for staged pilot groups and broader deployments alike, but it is usually unnecessary for a single one-off installation.

Action path

1. Install EventReporter on a reference system.
2. Configure and test the required services, rulesets, filters, and actions.
3. Export the configuration as a text-based registry file.
4. Package the EventReporter software and the exported configuration for your deployment tool.
5. Deploy to target systems by using your standard automation platform such as Group Policy, SCCM, PowerShell, or another software distribution tool.
6. Import the configuration on the target systems.
7. Start the EventReporter service and verify that event collection and forwarding work as expected.

Important notes

- Review machine-specific settings such as file paths, service accounts, credentials, and target addresses.
- Test on a non-production system before broader deployment.
- Keep the configuration export in text form so it can be reviewed and tracked.

Related information

- How to Copy the EventReporter Configuration to Other Servers
- Installation

How to Copy the EventReporter Configuration to Other Servers

Question

How can I copy an EventReporter configuration to other servers?

Answer

Export the configuration from one EventReporter system as a text-based registry file and import it on the target systems.

Details

This is useful when several servers should use the same rulesets, actions, and service structure. It avoids rebuilding the same configuration manually on each system.

Action path

1. Open the EventReporter Configuration Client on the source system.
2. Go to **Computer -> Export Settings to Registry File**.
3. Save the export as a text-based `.reg` file.
4. Copy the file to the target systems.
5. Import the file on each target system.
6. Open the EventReporter Configuration Client and review any system-specific values such as local file paths, credentials, or target hosts.
7. Apply the configuration and restart the EventReporter service if required.

Related information

- How Do I Perform a Repeatable Deployment of EventReporter?
- How to Export EventReporter Settings for a Support Call

How Do Remote Administration and Browser-Based Log Review Fit Together?

Question

How do remote administration and browser-based log review work in the current Adiscon Windows products?

Answer

Treat them as two separate functions.

- Administrative changes are made with the Configuration Client, either on the local system or through the remote-connect workflow when that product supports it.
- The current product family does not use a built-in browser administration interface for service configuration.
- Browser-based review is handled by Adiscon LogAnalyzer, which is a separate and optional component for stored data. It is not the service administration interface.

Details

Remote administration

The current product family uses the Windows Configuration Client for administration. Depending on the product and deployment style, you typically work in one of these ways:

- connect to another machine from the client
- export and import configuration
- use a repeatable deployment process for repeated deployments

This means remote administration is a client-and-deployment workflow, not a built-in browser administration console.

When the client connects to another machine directly, the target system must be reachable over the network and the current user must have the required access rights on that remote machine.

Browser-based log review

Adiscon LogAnalyzer is the browser-based component in this ecosystem. It is used to review data that has already been written to a file or database. It is deployed separately from the logging service and requires its own web server and PHP runtime.

That split is important:

- LogAnalyzer is for stored data review.
- The logging service still receives, processes, and stores or forwards data.
- The Configuration Client is still the place where you change service settings.

What this means operationally

If you want both remote administration and browser-based visibility, plan them as separate pieces:

1. Decide how the product configuration will be administered remotely.
2. Decide where retained data will be stored.
3. Deploy LogAnalyzer separately if browser-based review of stored data is needed.

Action path

1. Use the Configuration Client for administrative changes.
2. If the target system is remote, use the product’s remote-connect or deployment workflow.
3. Configure file or database storage for the data you want to review later.
4. Deploy Adiscon LogAnalyzer separately on a web server if browser-based review is required.
5. Point LogAnalyzer at the same stored data source and verify end-to-end that new rows appear there.

Related information

- `../tutorials/loganalyzer-setup-and-use`
- Use the product-specific remote-connect page when the Configuration Client can connect to another machine directly.
- Use the product-specific deployment or configuration-copy guidance when you need repeatable deployment across multiple systems.

How Can I Obtain a Printable Version of the EventReporter Manual?

Answer

You can use the online EventReporter manual in your browser or download a printable PDF version from the EventReporter manual site:

EventReporter Manual

Details

The online manual is usually the best choice for browsing, searching, and following links between related topics. If you need a printable copy for offline review, internal distribution, or change control, use the PDF version available on the manual site.

If EventReporter is already installed, local help is also included with the installation, so an additional download is often unnecessary.

Can EventReporter Write to a UNC Path?

Question

Can EventReporter write log files to a UNC path such as `\server\share\logs`?

Answer

Yes, but not with the default Local System account. To use a UNC path, run the EventReporter service under an account that has permission to access the target share.

Details

UNC path support itself is not the limiting factor. The important part is the Windows security context under which the service runs.

Action path

1. Create or choose a service account that has access to the target share.
2. Grant that account the required share and NTFS permissions.
3. Open the Windows Services console.
4. Open the properties of the EventReporter service.
5. Change the logon account from Local System to the chosen service account.
6. Restart the EventReporter service.
7. Configure the file action to use the UNC path.
8. Trigger a test event and verify that the target file is created or updated.

Related information

- File Logging Options
- Tutorial: Collect Windows Events and Write Them to a File

Which Database Format Should I Use with EventReporter?

Question

Which database format should I use with EventReporter, and what should I consider before logging events into a database?

Answer

Use the default EventReporter database format when you want the fastest supported setup or compatibility with the standard Adiscon table layout. Use a custom schema when EventReporter must integrate with an existing database design.

EventReporter is not limited to an “internal schema” path. The database action can write to supported ODBC-accessible databases with user-defined schemas, as long as you configure the table name and field mapping correctly.

For most production deployments, use a server-grade database such as Microsoft SQL Server, MySQL, MariaDB, or PostgreSQL. Avoid Microsoft Access for production logging.

Details

EventReporter can write events into an ODBC-accessible database through the Write to Database action. The built-in default format is the safest choice for first-time setup because it matches the fields that Adiscon tools expect and avoids unnecessary mapping work. However, the action is also a general integration feature: it can write to your own table and column layout when that is the real requirement.

Use the default format when:

- you are setting up database logging for the first time
- you want predictable field mapping
- you plan to use the built-in **Create Database** flow
- you plan to analyze the data with Adiscon-compatible or standard SQL tooling
- you do not have an existing schema that EventReporter must integrate with

Use a custom format only when:

- your organization already has a fixed schema
- another system requires specific column names or data types
- you have tested the mapping and understand the compatibility impact

What the database action does not do

The action is a writer and mapping layer. It does not design your schema, choose your indexes, or build your reporting model for you. If you choose a custom schema, you own the destination database design.

Action path

1. Decide whether you need the default supported schema or integration with an existing custom schema.
2. Create and test an ODBC **System DSN** on the EventReporter host.
3. In EventReporter, add a Write to Database action to the ruleset that should store events.
4. For the default schema path, follow Tutorial: Write Windows Events to Microsoft SQL Server.
5. For the custom integration path, follow Tutorial: Integrate EventReporter with a Custom Database Schema.
6. Trigger a test event and verify that a row is inserted into the intended table.

Related information

- ODBC Database Options
- Tutorial: Write Windows Events to Microsoft SQL Server
- Tutorial: Integrate EventReporter with a Custom Database Schema
- mariadb-odbc-support

Can EventReporter Work with Custom Event Logs?

Question

Can EventReporter monitor custom Windows event logs and application-defined logs?

Answer

Yes. EventReporter can monitor custom Windows event logs as long as the target system exposes them through the Windows Event Log infrastructure.

Details

Many Windows applications and server roles register their own event sources or custom logs. If Windows exposes those logs or channels, EventReporter can read them through the Event Log Monitor service.

The practical limitation is usually not EventReporter itself, but whether the log exists on the system and whether the account context used by the service can read it.

Action path

1. Identify the custom log or channel in Windows Event Viewer.
2. Configure an Event Log Monitor service to read that log or channel.
3. Bind the service to a ruleset with a visible action such as Write to File.
4. Apply the configuration.
5. Trigger a test event in the custom log and confirm that EventReporter captures it.

Licensing and purchasing

How Do I Enter EventReporter License Information?

Answer

Open the EventReporter Configuration Client, go to **General** -> **License**, enter the registration name exactly as provided, import the license key, save the configuration, and restart the EventReporter service.

Details

After you purchase EventReporter, Adiscon sends the license information by email. That message contains:

- the registration name
- the license key

The registration name is case-sensitive and must match the delivered value exactly.

For Event Log Monitor deployments, licensing is based on the source systems whose Windows Event Logs are collected or forwarded. A license is required for each monitored system, regardless of whether that source system is a physical server, a workstation, or a virtual machine.

Action path

1. Open the EventReporter Configuration Client.
2. In the left pane, expand **General** and select **License**.
3. Copy the registration name from the delivery email into **Registration Name**.
4. Copy the full license key and click **Import from Clipboard**.
5. Save the configuration.
6. Restart the EventReporter service so the updated license state is applied.

Related information

- Installation
- Edition Comparison
- How do I contact Adiscon sales?

Platform and compatibility

Do the configuration clients require .NET Framework, or is .NET Core or .NET 5+ enough?

Question

Can the Windows configuration clients run with only the newer .NET runtime installed, such as .NET Core, .NET 8, or .NET 10?

Answer

No. The Windows configuration clients require **Microsoft .NET Framework 4.7.2 or a newer .NET Framework 4.x release**.

`.NET Framework 4.x` is a Windows-only runtime family. `.NET Core` and `.NET 5+` are a different cross-platform runtime family.

`.NET 5+` continues the `.NET Core` line under a new name. It is not a newer version of `.NET Framework 4.x`.

Installing only `.NET Core` or `.NET 5+` does not satisfy a `.NET Framework` requirement.

Details

For these products, the requirement applies to the **Windows configuration client**. The background service is a separate component.

If you deploy only the background service on a target system, the Configuration Client is not required on that system. In that case, this `.NET Framework` requirement applies where the Configuration Client is installed and used, not to the service-only target.

The following satisfy this requirement:

- `.NET Framework 4.7.2`
- `.NET Framework 4.8`
- `.NET Framework 4.8.1`

The following do not satisfy this requirement on their own:

- `.NET Core 3.x`
- `.NET 5`
- `.NET 6`
- `.NET 8`
- `.NET 10`

Action path

1. Check whether the system has `.NET Framework 4.7.2` or a newer `.NET Framework 4.x` release installed.
2. If it is missing, install `.NET Framework` separately or allow the product installer to add the required Framework components.
3. If the system only has `.NET Core` or `.NET 5+`, do not assume that the configuration client can run.

Related information

- Microsoft documentation on .NET Framework versions and dependencies: https://learn.microsoft.com/en-us/dotnet/framework/install/versions-and-dependencies
- Microsoft documentation on installing .NET on Windows: https://learn.microsoft.com/en-us/dotnet/core/install/windows

Is EventReporter v19+ supported on Windows Server IoT 2025?

Overview

This FAQ answers whether EventReporter v19+ is supported on Windows Server IoT 2025 and outlines current guidance, functional status, and considerations for deployments, including Server Core.

Notes

Support Status

Official support:

- Windows Server IoT 2025 is not yet explicitly listed in the EventReporter v19+ platform matrix.

Functional status:

- EventReporter v19+ is known to function properly on Windows Server IoT 2025 (including Server Core) based on internal testing and field feedback.

Guidance for Server Core Deployments

Windows Server IoT 2025 Server Core does not provide a graphical user interface. For headless deployments, we recommend configuring EventReporter using Adiscon Config Files (`*.cfg`), a portable, file-based configuration format.

Recommended workflow:

1. Create the configuration on a GUI-enabled machine - Install EventReporter and open the Configuration Client - Configure rules, services, and actions as required - Export the configuration as Adiscon Config Files (`*.cfg`)
2. Transfer the configuration to Server Core - Copy the exported `.cfg` to the Server Core system (e.g., via PowerShell Remoting or SMB)
3. Enable File Config Mode and set paths via registry

   Registry path: `HKEY_LOCAL_MACHINE\SOFTWARE\Adiscon\EventReporter\Settings`

   Required values:

   - `szFileConfig` (REG_SZ): Example `c:\configs\eventreporter\central-server.cfg`
   - `szDataDirectory` (REG_SZ): Example `c:\configs\eventreporter\`
   - `iAccessMode` (REG_DWORD): `1` (enables file config mode)

> **Important**
>
> When running in file config mode, ensure the service account has read access to the configuration file and write access to the data directory.

Sales

This section answers sales and licensing questions for Adiscon products in a question-focused format.

Use this section for commercial topics (quotes, orders, licenses, renewals, and purchase orders). For technical troubleshooting and product support, use the FAQ and troubleshooting sections in the relevant product manual.

Quick actions

- I need to contact sales: How do I contact Adiscon sales?.
- I need to request a quote: What should I include in a quote request?.
- I opened a ticket and need to check delivery/notification behavior: What happens after I open a sales ticket?.
- I need help with purchase orders and billing data: How do purchase orders and billing requests work?.
- I need product licensing and ordering links: Licensing and ordering.

- I need to confirm offline or air-gapped licensing behavior: Air-gapped environments.
- I need UpgradeInsurance policy details: UpgradeInsurance.

How do I contact Adiscon sales?

Short answer

Use the Customer Service System as the primary channel; email to `sales@adiscon.com` is also accepted and creates tickets.

Question

How do I contact Adiscon sales for quotes, ordering, licensing, or renewals?

Answer

Use the Customer Service System. This is the primary contact path for sales questions.

Details

- You can also send email to `sales@adiscon.com`.
- Emails to `sales@adiscon.com` create tickets automatically.
- The ticket portal is recommended because email delivery can be unreliable in both directions (customer to Adiscon and Adiscon to customer).
- If you do not see a response, this is usually a delivery or filtering issue, not an intentionally unanswered request.
- Choosing a sales category helps speed up handling. If a ticket is opened in the wrong category, it is reassigned internally.
- For language handling, see **`../contact-language-policy`**.

Technical support routing

For technical incidents, troubleshooting, and product behavior issues, use the support category in the same system or the FAQ and troubleshooting sections in your product manual.

Next action

Open a sales ticket in the Customer Service System or send email to `sales@adiscon.com` (emails create tickets).

What should I include in a quote request?

Short answer

You can start with a short request, and sales will guide you through missing details; providing more context upfront can speed up the first quote.

Question

What information should I provide to get an accurate sales quote quickly?

Answer

You can start with a short message like “I want to buy your tool.” Sales will follow up and guide you through any missing details.

Details

If you want a faster first quote

- Product name
- Edition (if known) or note that you need help selecting one
- Number of licenses or systems/devices to be covered
- Billing country
- Commercial contact email

Optional but speeds finalization

- Deployment type (for example production, test, or mixed)
- License term and UpgradeInsurance requirements
- Preferred purchase method (online order or purchase order)
- Company legal name and billing contact details
- Target purchase timeline
- Existing license, renewal, or expansion context

EU VAT note

Customers in EU member states are normally charged VAT. If your organization is eligible for VAT exemption, provide your official VAT ID during ordering or invoicing so the billing team can apply the exemption.

Optional copy/paste template

If you already have details, this template can speed up quote processing:

```
Product:
Edition (or "help me choose"):
Scope (licenses/systems/devices):
Deployment type (production/test/mixed):
License term / UpgradeInsurance needs:
Billing country:
Company legal name:
Purchase method (online order/PO):
Target purchase date:
Existing license context (if any):
VAT ID (EU, if applicable):
Contact email:
```

Next action

Open a sales ticket in the Customer Service System or send email to `sales@adiscon.com` (emails create tickets).

What happens after I open a sales ticket?

Short answer

Your ticket is confirmed by email and then handled in the same ticket thread, with first response usually within one business day.

Question

What should I expect after opening a sales ticket?

Answer

Adiscon sales reviews and responds in the same ticket thread, which is the primary place to track status and replies.

Normal flow

1. You open a ticket (via portal or email).
2. An automatic confirmation email is sent (if a valid address is available).

3. A sales agent replies in the same ticket thread.
4. Follow-up questions and updates continue in that same thread.

Details

Notification behavior

When a ticket is opened, an automatic response is sent by email to the registered address (if available).

If no automatic response arrives

Do the following:

- Spam or junk folders
- Mail security gateways
- Internal email policy filters

Then sign in to the Customer Service System to confirm whether your ticket was created.

Avoid duplicate tickets

If a ticket already exists, continue in that ticket instead of opening a duplicate request. Duplicate tickets can slow handling and split context.

When replying by email, keep the same subject/thread whenever possible so updates remain attached to the original ticket.

Next action

Check and continue the existing ticket in the Customer Service System. If no ticket exists, open a new one in the portal or send email to `sales@adiscon.com`.

How do purchase orders and billing requests work?

Short answer

Use the Customer Service System for PO processing, billing changes, and invoice requests; for enterprise procurement flows, describe your process and Adiscon will adapt as much as possible.

Question

How do I handle purchase orders, invoicing, and billing-data updates?

Answer

Use the Customer Service System for purchase order processing and billing requests. Include complete company and billing details to avoid order delays.

Details

Typical flow

1. Open a sales ticket in the Customer Service System.
2. Share PO and billing details in that ticket.
3. Sales confirms details and clarifies open points.
4. Order and invoice handling continues in the same ticket thread.

For larger organizations with their own procurement flow, open a sales ticket and describe your process. Adiscon will adapt as much as possible.

Purchase orders

Helpful information to include at the start:

- PO number (if already issued)
- Company legal entity
- Billing address
- Billing contact
- Product and scope

For formal purchase order requirements and additional details, see: Adiscon purchase orders.

Billing and invoice requests

For invoice details, billing data updates, tax/VAT information, and related order handling, contact sales via the Customer Service System.

Use the Customer Service System for billing-data changes both before and after order placement.

Security and language

- Prefer the ticket portal (including attachments) over plain email when sending sensitive billing data.
- For language handling, see **`../contact-language-policy`**.

Common requests

The sales team can also handle:

- Invoice copy and invoice re-send requests
- Billing address/contact changes
- VAT/tax-related billing updates

Next action

Open a sales ticket in the Customer Service System or send email to `sales@adiscon.com` (emails create tickets).

Licensing and ordering

Short answer

Use product ordering pages for direct purchase or open a sales ticket for edition guidance and licensing clarification.

Question

How do I order Adiscon products, and where do I clarify licensing questions?

Answer

Order through the product-specific ordering pages or via purchase order. If edition or licensing details are unclear, open a sales ticket first.

Details

Trial period

WinSyslog, EventReporter, MonitorWare Agent, and rsyslog Windows Agent provide 30-day trial functionality after installation. After that period, a valid license is required for full-feature operation.

License agreement (EULA)

The End User License Agreement (EULA) is shown during setup. If you need a copy or have licensing questions, contact the Customer Service System.

Product-specific links

Product	Pricing and ordering
WinSyslog	WinSyslog ordering page
EventReporter	EventReporter ordering page
MonitorWare Agent	MonitorWare Agent ordering page
rsyslog Windows Agent	rsyslog Windows Agent ordering page

Choosing an edition

If you are unsure which edition fits your requirements, open a sales ticket in the Customer Service System. A short request is enough to start.

Product-specific licensing details

Licensing and counting rules can be product-specific. For now, use the sales ticket process to clarify product-specific licensing details. These details will be documented in dedicated product-specific sales pages.

Offline and licensing behavior

For common licensing and deployment questions, see these canonical answers:

- Air-gapped environments
- Offline installation and activation
- Online verification after activation
- Perpetual licenses and UpgradeInsurance

Purchase methods

You can place orders through each product ordering page or by purchase order. If you need a formal quote before placing the order, open a sales ticket in the Customer Service System.

Maintenance and renewal

For maintenance and renewal policy details, see UpgradeInsurance.

Purchase orders and billing

For purchase order handling and billing details, see How do purchase orders and billing requests work?.

Next action

Open a sales ticket in the Customer Service System or send email to `sales@adiscon.com` (emails create tickets).

Air-gapped environments

Short answer

Adiscon products can be deployed and operated in fully closed environments without internet access.

Question

Can Adiscon products be used in completely closed or air-gapped environments?

Answer

Yes. Adiscon products support deployment and operation in fully closed networks.

Details

Products can be installed and operated in environments where internet access is not available or not permitted. They do not require outbound internet access for normal runtime operation in such environments.

If you also need details about license activation without internet access, see Offline installation and activation.

Action path

If your procurement or security process requires a written confirmation for a specific environment, open a sales ticket in the Customer Service System.

Related information

- Offline installation and activation
- Online verification after activation

Offline installation and activation

Short answer

Installation and license activation do not require internet access.

Question

Do Adiscon product installation and license activation require internet access?

Answer

No. Installation and license activation can be completed without an internet connection.

Details

Adiscon supports installation and license activation in offline environments. This allows products to be deployed in restricted networks where direct internet connectivity is not available.

This question is separate from whether the licensed product later requires online verification. For that, see Online verification after activation.

Action path

If you need product delivery or license handling guidance for an offline rollout, open a sales ticket in the Customer Service System.

Related information

- Air-gapped environments
- Online verification after activation

Online verification after activation

Short answer

After activation, Adiscon product licenses do not require ongoing internet connectivity, online heartbeats, or “phone home” checks to remain valid.

Question

Do Adiscon product licenses require online verification after activation?

Answer

No. After activation, the license remains valid without periodic online verification.

Details

Once the product is installed and the license has been applied, no ongoing outbound connection to Adiscon systems is required to keep the license active. There is no periodic online verification, heartbeat, or “phone home” mechanism needed to maintain licensed status.

This question is separate from whether installation and activation themselves require internet access. For that, see Offline installation and activation.

Action path

If your security or compliance process requires explicit confirmation about offline license behavior, open a sales ticket in the Customer Service System.

Related information

- Air-gapped environments
- Offline installation and activation
- Perpetual licenses and UpgradeInsurance

Perpetual licenses and UpgradeInsurance

Short answer

Adiscon product licenses are perpetual. UpgradeInsurance is a separate, time-limited service that is strongly recommended for access to major upgrades.

Question

Are Adiscon product licenses perpetual, and how does that differ from UpgradeInsurance?

Answer

Adiscon product licenses are perpetual. UpgradeInsurance is an additional, time-limited service for upgrade and maintenance benefits.

Details

A perpetual license allows continued use of the licensed product version after activation. It does not depend on periodic online verification to remain valid.

UpgradeInsurance is separate from the base license. It is a time-limited additional service that is strongly recommended if you want access to future major product versions and related maintenance benefits.

When UpgradeInsurance expires, the licensed product version does not stop working. What changes is entitlement to later major upgrades and the additional benefits covered by active UpgradeInsurance.

For detailed renewal and reinstatement policy, see UpgradeInsurance.

Action path

If you need confirmation for purchasing, renewal, or compliance review, open a sales ticket in the Customer Service System.

Related information

- UpgradeInsurance
- Online verification after activation

UpgradeInsurance

Short answer

UpgradeInsurance provides major-version upgrade rights and priority support while active; if coverage has lapsed, reinstatement is possible with back-dated start rules.

Question

What is UpgradeInsurance, and how do renewal and reinstatement work?

Answer

UpgradeInsurance is Adiscon's maintenance plan.

Details

What it includes

- Free major version upgrades during active coverage
- Priority support handling

Coverage period

UpgradeInsurance is available for periods between 1 and 5 years.

Coverage continuity and reinstatement

- UpgradeInsurance should be kept active through timely renewal.
- If coverage lapses, reinstatement is possible.
- If UpgradeInsurance was not purchased with the original license, it can still be added later.
- Reinstatement is back-dated to the previous coverage end date or, if it was never purchased, to the original license purchase date.

Difference from the base license

UpgradeInsurance is separate from the base product license. For the distinction between perpetual licenses and this time-limited service, see Perpetual licenses and UpgradeInsurance.

Examples

1. If coverage ended on June 30 and renewal is requested in September, reinstatement starts from June 30.
2. If no UpgradeInsurance was purchased with the initial order, later activation is back-dated to the original license purchase date.

How to request renewal or reinstatement

Open a sales ticket in the Customer Service System or send email to `sales@adiscon.com` (emails create tickets).

Next action

Open a sales ticket in the Customer Service System or send email to `sales@adiscon.com` (emails create tickets).

Reference

Use this section for lookup material, not for first-time setup. It collects the pages you typically need while operating, troubleshooting, or fine-tuning an existing EventReporter deployment.

Use `Getting Started` for first-time setup, `Tutorials` for task guidance, and `Configuration` for the main product setup pages.

Product controls and local operation

EventReporter Shortcut Keys

Use shortcut keys as an alternative to the mouse when working in EventReporter Client. Keyboard shortcuts may also make it easier for you to interact with EventReporter. All these shortcut keys are usually available in textboxes only. Listed below are the available short keys:

CTRL+S = Save

CTRL+X = Cut

CTRL+C = Copy

CTRL+V = Paste

CTRL+Z = Undo

Note: This is in synchronization with most major Windows applications.

Command Line Switches

There are several command line switches available for using EventReporter from the command line. To use these switches you need administrative rights.

- `-h` Show command line help
- `-v` Show version information and whether the service is installed
- `-i` Install service
- `-u` Remove (uninstall) service
- `-i <CustomServiceName>` Install service with a custom service name
- `-u <CustomServiceName>` Uninstall a service with a custom service name
- `-r` Run as console application
- `-r -o` Run once as console application

If you install the service, you can start and stop it with commands such as `net start`, `net stop`, `sc start`, `sc stop`, or PowerShell (`Start-Service` / `Stop-Service`). By using the `-r` switch, you run it only on the command line. When you close the command line, the program will stop working.

The `-v` switch gives you information about the version of the service.

Custom service name examples:

- `evntslog.exe -i CustomServiceName`
- `evntslog.exe -u CustomServiceName`

You can import Adiscon Config Format (cfg) configuration files via the command line as well. The syntax is quite easy. Simply execute the EventReporter configuration client and append the name of the configuration file.

Sample:

CFGEvntSLog.exe example.cfg

or

CFGEvntSLog.exe “example.cfg”

After this is executed, you will see the splash screen of the configuration client and then the import dialogue, which you have to confirm manually.

For doing a silent import, the `/f` parameter has to be appended. This will look like this:

```
CFGEvntSLog.exe "example.cfg" /f
```

Edition Comparison

Use the EventReporter edition comparison page when you need to confirm which services, actions, and product capabilities are available in each edition.

For licensing scope, count each source system whose Windows Event Logs are collected or forwarded through Event Log Monitor. This applies to physical systems and virtual machines alike.

EventReporter edition comparison

Technical lookup material

Comparison of properties

Available in MonitorWare Agent, EventReporter and WinSyslog

The property replacer is a reference - the actual properties are very depending on the edition purchased. We have just included information on what is available in which products for your ease and convenience.

Properties Available	MonitorWareAgent	WinSyslog	EventReporter
Standard Property	Yes	Yes	Yes
Windows Event Log	Yes		Yes
Syslog Message	Yes	Yes	
Disk Space Monitor	Yes		
File Monitor	Yes		
Windows Service Monitor	Yes		Yes
Ping Probe	Yes		
Port Probe	Yes		
Database Monitor	Yes		
Serial Port Monitor	Yes		
MonitorWare Echo Request	Yes		
System	Yes	Yes	Yes
Custom	Yes	Yes	Yes
NNTP Probe	Yes		
HTTP Probe	Yes		
FTP Probe	Yes		
SMTP Probe	Yes		
POP3 Probe	Yes		

Event Properties

Events have certain properties, for example the message associated with the event or the time it was generated. Each of this properties has an assigned name. The actual properties available depend on the type of event. The following sections describe both how to access properties as well as properties available.

Knowing about event properties is important for building complex filter conditions, customized actions as well as for integrating into a third-party system. Event properties provide a generic way to look at and process the events generated. Thus we highly recommend that you at least briefly read this reference section.

Accessing Properties

Properties are accessed by their name. The component used for this is called the “property replacer”. It is a generic component that allows you to merge properties from the event processed to e.g. the email subject line or a log file line. It is a central component that is used as often in the product as possible. The idea behind the property replacer is that there is often need to specify a value from the event processed.

The property replacer provides very powerful ways to access the properties: they cannot only be accessed as one full property. They can also be accessed as substrings and even be reformatted. As such, the property replacer provides a specific syntax to access properties:

`%property:fromPos:toPos:options%`

The percent-signs (“%”) indicate the start of a special sequence. The other parameters have the following meanings

FromPos and ToPos can be used to copy a substring from a lengthy property. The options allow to specify some additional formatting.

Within the properties, all time is based on UTC regardless if your preferred time is UTC or localtime. So if you want to display localtime instead of UTC, you have to use the following syntax: `%variable:::localtime%`

Property

This is the name of the property to be replaced. It can be any property that a given event possesses. If a property is selected that is empty for the event processed, an empty string is returned.

A property is either an event property, a custom property, a dynamic property or a system property.

If a property is selected that is not present, the result will always be an empty string, no matter which other options have been selected.

FromPos

If you do not want to use the full string from the property, you can specify a start position here. There are two ways to specify the start location:

Fixed Character position

If you know exactly on which position the string of interest begins, you can use a fixed location. In this case, simply specify the character position containing the first character of interest. Character positions are counted at 1.

Search Pattern

A search pattern is specified as follows:

/<search-pattern>/<options>

If a search pattern is specified, the property value is examined and the first occurrence of <search-pattern> is detected. If it is not found, nothing is returned. If it is found, the position where the pattern is found is the start position or, if the option “$” is specified, the position immediately after the pattern.

The search pattern may contain the “?” wildcard character, which represents any character. Other wildcards are not supported with the property replacer.

Please note that a slash inside the search pattern will terminate the search field. So pure slashes cannot be used. However, they can be escaped by prefixing them with a backslash (). The same applies to the ‘?’ character. For example, if you intend to search for `"http://"` inside a search pattern, you must use the following search string: `"/http:////"`.

Default Value

If the FromPos is not specified, the property string is copied starting at position 1.

ToPos

If you do not want to use the full string from the property, you can specify the highest character position to be copied here.

Absolute Position

Specify a simple integer if you would like to specify an absolute ending position.

Relative Position

This is most useful together with the search capabilities of FromPos. A relative position allows you to specify how many characters before or after the FromPos you would like to have copied. Relative positions are specified by putting a plus or minus (“+”/”-”) in front of the integer.

Please note: if you specify a negative position (e.g. -20), FromPos and ToPos will internally be swapped. That is the property value will not be (somehow) reversely copied but they will be in right order. For example, if you specify `%msg:30:-20%` actually character positions 10 to 30 will be copied.

Search Pattern

Search pattern support is similar to search pattern support in FromPos.

A search pattern is specified as follows:

/<search-pattern>/<options>

If a search pattern is specified, the property value is examined and the first occurrence of <search-pattern> is detected. The search is only carried out in the string that follows FromPos. If the string is not found, nothing is returned. If it is found, the position where the pattern is found is the ending position or, if the option “$” is specified, the position immediately after the pattern.

The search pattern may contain the “?” wildcard character, which represents any character. Other wildcards are not supported with the property replacer.

Please note that a slash inside the search pattern will terminate the search field. So pure slashes cannot be used. However, they can be escaped by prefixing them with a backslash (). The same applies to the ‘?’ character. For example, if you intend to search for `"http://"` inside a search pattern, you must use the following search string: `"/http:////"`.

Search Example

A common use case is to combine searches in ToPos and FromPos to extract a substring that is delimited by two other strings. To do so, use search patterns in both fields. An example is as follows: assume a device might generate message in the form `"... error XXX occurred..."` where “…” represents additional message text and XXX the actual error cause. You would like to extract the phrase `"error XXX occurred"`. To do so, use the following property replacer syntax: `%msg:/error/:/occurred/$/%`

Please note that the FromPos is used without the $-option, while in ToPos it is used. If it hadn’t been used in ToPos, only the part “error XXX “ would have been extracted, as the ToPos would point to the last character before the search string.

Similarly, if only “ XXX “ should be extracted, the following syntax might be used:

```
%msg:/error/$:/occurred/%
```

If you would also like to remove the spaces (resulting in just “XXX”), you must include them into the search strings:

```
%msg:/error /:/ occurred/$/%
```

Default

If not specified, the ending position will be the last character.

Options

Options allow you to modify the contents of the property. Multiple options can be set. They are comma-separated. If conflicting options are specified, always the last option will be in effect (e.g. specifying “uppercase,lowercase” will lead to lowercase conversion of the property value).

The following options are available with this release of the product:

lowercase

All characters in the resulting property extract will be converted to lower case.

uppercase

All characters in the resulting property extract will be converted to upper case.

uxTimeStamp

This is a special switch for date conversions. It only works if the extracted property value is an ISO-like timestamp (YYYY-MM-DD HH:MM:SS). If so, it will be converted to a Unix-like ctime() timestamp. If the extracted property value is not an ISO-like timestamp, no conversion happens.

uxLocalTimeStamp

This is the same as uxTimeStamp, but with local time instead of GMT.

date-rfc3339

This option is for replacing the normal date format with the date format from RFC3339.

date-rfc3164

This option is for replacing the normal date format with the date format from RFC3164.

date-rfc3164strict

Does the same as date-rfc3164 but when the date is below 10, two spaces will be added between Month and day (Which is defined in rfc3164).

escapecc

Control characters* in property are replaced by the sequence `##hex-val##`, where* hex-val is the hexadecimal value of the control character (at least two digits, may be more).

spacecc

Control characters* in the property are replaced by spaces. This option is most* useful when a message contains control characters (e.g. a Windows Event Log Message) and should be written to a log file.

compressspace

Compresses multiple consecutive space characters into a single one. The result is a string where all words are separated by just single spaces. To also compress control characters, use the compressspace and spacecc options together (e.g.``'%msg:::spacecc,compressspace%'). Please note that space compression happens on the final substring. So if you
use the FromPos and ToPos capabilities the substring is extracted first and
then the space compression applied. For example, you may have the msg string"1 2". There are two space between 1 and 2. Thus, the property replacer expression:``%msg:1:3:compressspace%

will lead to `"1  "` ('1' followed by two spaces). If you intend to receive
`"1 2"``('1' followed by one space, followed by '2'), you need to use ``%msg:1:4:compressspace%`

or

`%msg:1:/2/$:compressspace%`

In the second case, the exact length of the uncompressed string is not known, thus a search is used in topos to obtain it. The result is then space-compressed.

compsp

Exactly the same as compressspace, just an abbreviated form for those that like it brief.

csv

For example `%variable:::csv%`. This option will create a valid CSV string. For example a string like this:`this is a "test"!` becomes this `"this is a ""test""!"` where quotes are replaced with double quotes.

cef

Convert string content into valid McAfee CEF Format. This means that `=``will be replaced with=``` and `\` will be replaced with `\`. **convgermuml**

Converts German Umlaut characters to their official replacement sequence (e.g. "ö" –> "oe")

localtime

Now you can print the Time with localtime format by using``%variable:::localtime%``

nomatchblank

If this is used, the Property Replacer will return an empty string if the frompos or topos is not found.

replacepercent

This option replaces all `%` occurrences with a double `%%`, which is needed for the property replacer engine in case that a string is reprocessed. This is needed because the percent sign is a special character for the property replacer. Once the property is processed, the double``%%``become automatically one``%``. **toipv4address**

Property string will be converted into IPv4 Address format if possible.

toipv6address

Property string will be converted into IPv6 Address format if possible.

crlftovbar

Does the same as date-rfc3164 but when the date is below 10, two spaces will be added between Month and day (Which is defined in rfc3164).

removecc

Removes all control characters from 0x00 to 0x1F

replacechar

Replaces a single character with another single character.

How ASCII characters are being handled:

Sample:`%msg:$x:$y:replacechar%`

Broken down:

```
%msg:$``<- Tells property replacer that a character is being expected (At the moment only for R
EPLACECHAR Option). ``x``<- The character to search for ``:
```

```
$``<- Tells property replacer that a character is being expected (At the moment only for REPLAC
ECHAR Option). ``y``<- The character to replace with ``:
```

```
replacechar%
```

How special characters are handled?

Sample: `%msg:$\n:$|:replacechar%`

`%msg:$` <- Tells property replacer that a character is being expected (At the moment only for REPLACECHAR Option). `\n` <- The character to search for special character, possible values: t for tab,

n for newline,

v for verticaltab,

f for formfeed,

r for carriage return

for an actual backslash.``:``

```
$``<- Tells property replacer that a character is being expected (At the moment only for REPLAC
ECHAR Option). ``|``<- The character to replace with ``:
```

```
replacechar%
```

`*` = control characters like e.g. carriage return, line feed, tab, …*

Important: All option values are case-sensitive. So “uxTimeStamp” works while “uxtimestamp” is an invalid option!

Simple Examples

A good example for this is the email subject line, which has severe length constraints. If you would like to have only the first 40 characters of the actual message text in the subject, you could use the replacer: `"%msg:1:40%"`. If you know the first 10 characters of the message are meaningless for you but you would like to see the full rest of the message (no matter how long it may be), you can use a sequence like `"%msg:11%"`.

If you would just like to see the plain message from beginning to end, you can simply omit frompos and topos: `"%msg"`. Of course, all of these sample not only work with the “msg” property, but also with all others like “facility”, or “priority”, or W3C-log header extracted property names.

More complex Examples

If you would like to extract the 50 characters from the message after the word DROP, you would use the following replacer string:`%msg:/DROP/$:+50%`

If you would like to have the first 40 characters in front of the string “- aborted” (including that string):

```
%msg:/- aborted/$:-40%
```

If you would like to receive everything starting from (and including) “Log:”:

```
%msg:/Log/%
```

If you would like to have everything between the string “FROM” and “TO” including NONE of the both searchstrings:

```
%msg:/FROM/$:/TO/%
```

If you would just like to log lowercase letters in your log messages:

```
%msg:::lowercase%
```

And if you would just like to have the first 50 characters (and these in lower case):

```
%msg:50:::lowercase%
```

If you need to change a timestamp to a UNIX-like timestamp, you could use this:

```
%datereceived:::uxTimeStamp%
```

Please see also the focused sample in the topos description.

A real world Sample

We use the following template to generate output suitable as input for MoniLog:

```
%timegenerated:1:10%,%timegenerated:12:19%,%source%,%syslogfacility%,%syslogprio rity%,EvntSlog
: %severity% %timereported:::uxTimeStamp%: %source%/%sourceproc% (%id%) - "%msg%"%$CRLF%
```

Please note: everything is on one line with no line breaks in between. This example is from the “write to file” action (with custom file format).**

System Properties

System properties are special sequences that can be helpful. They are available with all event types. They are:

$CRLF

A Windows newline sequence consisting in the characters CR and LF. If you just need CR, you can use `%$CRLF:1:1%` and if you need use LF you can use `%$CRLF:2:2%`

$TAB

An US-ASCII horizontal tab (HT, 0x09) character

$HT

same as $TAB

$CR

A single US-ASCII CR character (shortcut for `%$CRLF:1:1%`) **$LF**

A single US-ASCII LF character (shortcut for``%$CRLF:2:2%``) **$xNN**

A single character, whose value (in hexadecimal) is given by NN. NN must be two hexadecimal digits - a leading zero must be used if a value below 16 is to be represented. The value 0 (%x00) is invalid and - if specified - replaced by the “?” character.

As an example, $CR could also be expressed as %$x0d%.

Please note that only one character can be represented. If you need to specify multiple characters, you need multiple $xNN sequences. An example may be $CRLF which could also be specified as %$x0d%%$x0a% (but not as %$x0d0a%).

$NOW

Contains the current date and time in the format:`YYYY-MM-DD HH.MM.SS`

Please note that the time parts are delimited by `'.'` instead of `':'`. This makes the generated name directly suitable for file name generation.

If you need just parts of the timestamp, please use the property replacer’s substring functionality to obtain the desired part. Use``%$NOW:1:4% to get the year,``

`%$NOW:6:7% to get the month,`

…

`%$NOW:1:10% to get the full datestamp,`

`%$NOW:12:20% to get the full timestamp`

$NEWUUID

Creates a new UUID (Universally Unique Identifiers), a unique 128-bit integer represented as a 32 digit hexadecimal number.

Custom Properties

Users can create an unlimited number of custom properties. These can be created with for example the “PostProcess” action (if the product edition purchased supports this action).

Custom properties can theoretically have any name, but Adiscon highly recommends to prefix them with “u-” (e.g. “u-MyProperty” - “u” like “user”). This ensures that no compatibility problems will arise in current and future versions of the software. Adiscon guarantees that it will never use the “u-” prefix for Adiscon-assigned properties.

Custom properties can be used just like regular properties. Wherever you can specify a property, you can also specify a custom property.

Event-Specific Properties

Each network event is represented by a so-called “Event Record” (sometime also named an “InfoUnit”, an “Unit of Information”). Data obtained from all services will end up as an event. For example, Windows Event Log data, syslog data, and a file line obtained by the file monitor will all be an event. That kind of generalization make it easy to deal with all of these events in a consistent way.

Each event has a set of properties which in turn have values. For example, there is a property named “source” and it will always contain an indication of which system the event originated on. Obviously, not every event source does support all properties. For example, a syslog message does not contain a Windows Event ID - simply because there is no such thing as an event ID in syslog. So, depending on the type of event, it may contain different properties.

In order to make the product really generally useful, some few properties have been defined in a generic way and are guaranteed to be present in every event, no matter what type it may have. Sometimes this is a “natural” common property, like the “fromhost”. Sometimes, though, it may look a bit artificial. An example of the later is the “syslogfacility” property. It is guaranteed to be present in every event - but actually this is a syslog-only thing. The non- syslog event sources either emulate this property (in a consistent manner) or allow the user to configure a syslogfacility that should be used for all events generated by that service. At the bottom line, this will ensure that the property is available in all events and - given proper configuration - that can be extremely helpful for the administrators to set up things in a powerful and generic way.

Standard Properties

As outlined under Event Properties, these are properties present in all types of events. Some event types have only these standard properties. Others have additional properties. Those with additional properties are documented in the other sections. If there is no specific documentation for a specific event type, this means that it supports the standard properties, only.

msgPropertyDescribed

A human-readable representation of the message text. While this is generally available, the exact contents largely depends on the source of the information. For example, for a file monitor it contains the file line and for a syslog message it contains the parsed part of the syslog message.

source

The source system the message originated from. This can be in various representations (e.g. IP address or DNS name) depending on configuration settings.

localhostname

On service startup it is automatically set to the local system computer name. It is read only and can be used if source property is not usable. E.g. if the Source property cannot be translated to IP format because the event log entry was recorded with an old computer name that no longer exists.

resource

A user-assigned numerical value. Does not have any specific meaning. Primarily intended for quick filtering.

CustomerID

A user-assigned numerical value. Does not have any specific meaning. Primarily intended for quick filtering.

SystemID

A user-assigned numerical value. Does not have any specific meaning. Primarily intended for quick filtering.

timereported

The time the originator tells us when this message was reported. For example, for syslog this is the timestamp from the syslog message (if not configured otherwise). Please note that timereported eventually is incorrect or inconsistent with local system time - as it depends on external devices, which may not be properly synchronized.

For Windows Event Log events, timereported contains the timestamp from the event log record.

timegenerated

The time the event was recorded by the service. If messages are forwarded via SETP, this timestamp remains intact.

importance

Reserved for future use.

iut

Indicates the type of the event. Possible values are:

```
1- syslog message
2- heartbeat
3- Windows Event Log Entry
4- SNMP trap message
5- file monitor
8- ping probe
9- port probe
10- Windows service monitor
11- disk space monitor
12- database monitor
13- serial device monitor
```

iuvers

Version of the event record (info unit). This is a monitorware internal version identifier.

Windows Event Log Properties

id

Windows Event ID

severity

severity as indicated in the event log. This is represented in string form. Possible values are:

```
[INF] - informational
[AUS] - Audit Success
[AUF] - Audit failure
[WRN] - Warning
[ERR] - Error
[NON] - Success (called "NON" for historical reasons)
```

severityid

The severity encoded as a numerical entity (like in Windows API)

sourceproc

The process that wrote the event record (called “source” in Windows event viewer).

category

The category ID from the Windows Event Log record. This is a numerical value. The actual value is depending on the event source.

catname

The category name from the Windows Event Log record. This is a string value. The actual value is depending on the event source. This value is a textual representation from the Category ID. This property could contain line feeds, which can be removed by activating the option “Remove Control Characters from String Parameters” in the advanced options of the EventLog Monitor Service.

user

The user name that was recorded in the Windows Event Log. This is “NA” if no user was recorded.

NTEventLogType

The name of the Windows Event Log this event is from (for example “System” or “Security”).

bdata

Windows Event Log records sometimes contain binary data. The Event Log Monitor service can be set to include this binary data into the event, if it is present. If it is configured to do so, the binary data is put into the “bdata” property. Every byte of binary data is represented by two hexadecimal characters.

Please note that it is likely for bdata not to be present. This is because the binary data is seldom used and very performance-intense. (%id%) - “%msg%”%$CRLF%

Windows Event Log V2 Properties

id

Windows Event ID

severity

severity as indicated in the event log. This is represented in string form. Possible values are:

```
[INF] - informational
[AUS] - Audit Success
[AUF] - Audit failure
[WRN] - Warning
[ERR] - Error
[NON] - Success (called "NON" for historical reasons)
```

severityid

The severity encoded as a numerical entity (like in Windows API)

sourceproc

The process that wrote the event record (called “source” in Windows event viewer).

category

The category ID from the Windows Event Log record. This is a numerical value. The actual value is depending on the event source.

catname

The category name from the Windows Event Log record. This is a string value. The actual value is depending on the event source. This value is a textual representation from the Category ID. This property could contain line feeds, which can be removed by activating the option “Remove Control Characters from String Parameters” in the advanced options of the EventLog Monitor Service.

user

The user name that was recorded in the Windows Event Log. This is “NA” if no user was recorded.

nteventlogtype

The name of the Windows Event Log this event is from (for example “System” or “Security”).

channel

The channel property for event log entries, for classic Event logs they match the `%nteventlogtype%` property, for new event logs, they match the “Event Channel”.

sourceraw

This contains the full internal name of the event source for new event logs, for classic event logs it contains the same value as in %sourceproc%.

level

Textual representation of the event log level (which is stored as a number in `%severityid%`). This property is automatically localized by the system.

categoryid

Internal category id as number.

keyword

Textual representation of the event keyword. This property is automatically localized by the system.

user_sid

If available, contains the raw SID of the username (`%user%`) property.

recordnum

Contains the internal event record number. Please note that if the event log has been truncated before, it may not start with 0 or 1 but a higher number.

Syslog Message Properties

rawsyslogmsg

The message as it was received from the wire (unparsed).

syslogfacility

The facility of a syslog message. For non-syslog messages, the value is provided based on configuration. In essence, this is simply an integer value that can be used for quick filtering inside your rules.

syslogfacility_text

The facility of a syslog message. This property is automatically created by using the syslogfacility properly and set to these values: `"Kernel", "User", "Mail", "Daemons", "Auth", "Syslog", "Lpr", "News", "UUCP", "Cron", "System0", "System1", "System2", "System3", "System4", "System5", "Local0", "Local1", "Local2", "Local3", "Local4", "Local5", "Local6", "Local7"`

syslogpriority

The severity of a syslog message. For non-syslog messages, this should be a close approximation to what a syslog severity code means.

syslogpriority_text

The severity of a syslog message. This property is automatically created by using the syslogpriority properly and set to these values:
`"Emergency", "Alert", "Critical", "Error", "Warning", "Notice", "Informational", "Debug"`

syslogtag

The syslog tag value, a short string. For non-syslog messages, this is provided based on configuration. In most cases, this is used for filtering.

syslogver

Contains the syslog version number which will be one or higher if a rfc 5424 valid message has been received, or 0 otherwise

syslogappname

Contains the appname header field, only available if the Syslog message was in rfc 5424 format. Otherwise, this field will be emulated by the %syslogtag% property

syslogprocid

Contains the procid header field, only set if the Syslog message was in rfc 5424 format.

syslogmsgid

Contains the msgid header field, only set if the Syslog message was in rfc 5424 format.

syslogstructdata

Contains the structdata header field (in raw format), only set if the Syslog message was in rfc 5424 format.

syslogprifac

Contains combined syslog facility and priority useful to build your own custom syslog headers

Disk Space Monitor

currusage

The currently used disk space.

maxavailable

The overall capacity of the (logical) disk drive.

CPU/Memory Monitor

wmi_type

This variable is a string and can be one of the following variables: cpu_usage, mem_virtual_usage, mem_physical_usage, mem_total_usage.

cpu_number

Number of the current checked CPU.

cpu_load

The workload of the CPU as number, can be 0 to 100.

mem_virtual_load

How much virtual memory is used (MB).

mem_virtual_max

How much virtual memory is max available (MB).

mem_virtual_free

How much virtual memory is free (MB).

mem_physical_load

How much physical memory is used (MB).

mem_physical_max

How much physical memory is max available (MB).

mem_physical_free

How much physical memory is free (MB).

mem_total_load

How much total(Virtual+Physical) memory is used (MB).

mem_total_max

How much total(Virtual+Physical) memory is max available (MB).

mem_total_free

How much total(Virtual+Physical) memory is free (MB).

File Monitor

genericfilename

The configured generic name of the file being reported.

generatedbasefilename

Contains the generated file name without the full path.

Special IIS LogFile Properties

The Logfile Fields in IIS Logfiles are customizable, so there is no hardcoded command for their use.

The property-name depends on its name in the logfile. For example we take this Logfile:

```
#Software: Microsoft Internet Information Services 5.0
#Version: 1.0
#Date: 2005-10-27 14:15:25
#Fields: date time c-ip cs-username s-ip s-port cs-method cs-uri-stem
cs-uri-query sc-status cs(User-Agent)
2005-10-27 14:15:16 127.0.0.1 - 192.168.0.1 443 POST /eCommerce/asdf.php
2005-10-27 14:15:16 127.0.0.1 - 192.168.0.1 443 POST /eCommerce/asdf.php
2005-10-27 14:15:16 127.0.0.2 - 192.168.0.1 443 POST /eCommerce/asdf.php
2005-10-27 14:15:16 127.0.0.2 - 192.168.0.1 443 POST /eCommerce/asdf.php
```

As you can see, in our sample the fields are named: date, time, c-ip, cs-username, s-ip, and so on.

To use them as a Property inside our MonitorWareProducts, just use the names from your Logfile and add a “p-” before it:

p-date

The Date on which the Event occurs

p-time

The Time on which the Event occurs

p-c-ip

The IP address of the User which accessed

p-cs-username

The Username of the User which accessed

p-s-ip

The Server IP

p-s-port

The Server Port

p-cs-method

The Client-Server Method (POST,GET)

p-cs-uri-stem

The accessed File including its path

Windows Service Monitor

sourceproc

The name of the service whose status is being reported (from the Windows service registry).

Ping Probe

echostatus

Status returned for the echo request

The status value can be one of the following:

```
0 = IP_SUCCESS
11002 = IP_DEST_NET_UNREACHABLE
11003 = IP_DEST_HOST_UNREACHABLE
11010 = IP_REQ_TIMED_OUT
11013 = IP_TTL_EXPIRED_TRANSIT
11016 = IP_SOURCE_QUENCH
11018 = IP_BAD_DESTINATION
```

roundtriptime

Round trip time for the ping packet (if successful)

Port Probe

responsestatus

The status of the probe.

responsemsg

The response message received (if any)

Database Monitor

Database-Monitor created events are a bit different than other events. The reason is that the database fields themselves become properties - but obviously these are not fixed but depend on what you monitor.

All queried data fields are available as properties via their database field name **prefixed with “db-”**.

An example to clarify: we assume the following select statement is used for the database monitor:

select name, street, zip, city from addresses

There is also an ID column named “ID”. So the event generated by this database monitor will have the following specific properties:

- db-ID
- db-name
- db-street
- db-zip
- db-city

These properties will contain the field values as they are stored in the database. Please note that NULL values are translated into empty strings (“”), so there is no way to differentiate a NULL value from an empty string with this version of the database monitor.

Other than the custom “db-” properties, no specific database monitor properties exist.

Serial Monitor

portname

The name of the port that the data originated from (typical examples are COM1, COM2). The actual name is taken from the configuration settings (case is also taken from there).

MonitorWare Echo Request

responsestatus

The status of the echo request. Possible values:

```
0 - request failed (probed system not alive)
1 - request succeeded
```

If the request failed, additional information can be found in the * msg* standard property.

FTP Probe

ftpstatus

The status of the connection.

ftprespmsg

The response of the connection.

IMAP Probe

imapstatus

The status of the connection.

imaprespmsg

The response of the connection.

NNTP Probe

nntpstatus

The status of the connection.

nntprespmsg

The response of the connection.

SMTP Probe

smtpstatus

The status of the connection.

smtprespmsg

The response of the connection.

POP3 Probe

pop3status

The status of the connection.

pop3respmsg

The response of the connection.

HTTP Probe

httpstatus

The status of the connection.

httprespmsg

The response of the connection.

Complex Filter Conditions

The rule engine uses complex filter conditions.

Powerful boolean operations can be used to build filters as complex as needed. A boolean expression tree is graphically created. The configuration program is modeled after Microsoft Network Monitor. So thankfully, many administrators are already used to this type of Interface. If you are not familiar with it, however, it looks a bit confusing at first. In this chapter, we are providing some samples of how boolean expressions can be brought into the tree.

Example 1

In this example, the message text itself shall be checked. If it contains at least one of three given strings, the filter should become true. If none of the string is found, the boolean expression tree evaluates to false, which means the associated action(s) will not be executed.

In pseudo-code, the filter could be written like this:

```
If (msg  = "DUPADDRES") OR (msg = "SPANTREE") OR (msg = "DUPLEX_MISMATCH) then
    execute action(s)
end if
```

Please note: in the example, we have abbreviated “message” to just “msg”. Also note that for brevity reasons we use the equals (“=”) comparison operator, not the contains. The difference between the equals and the contains operator is that with “contains”, the string must just be part of the message.

In the filter dialog, this pseudo code looks as follows:

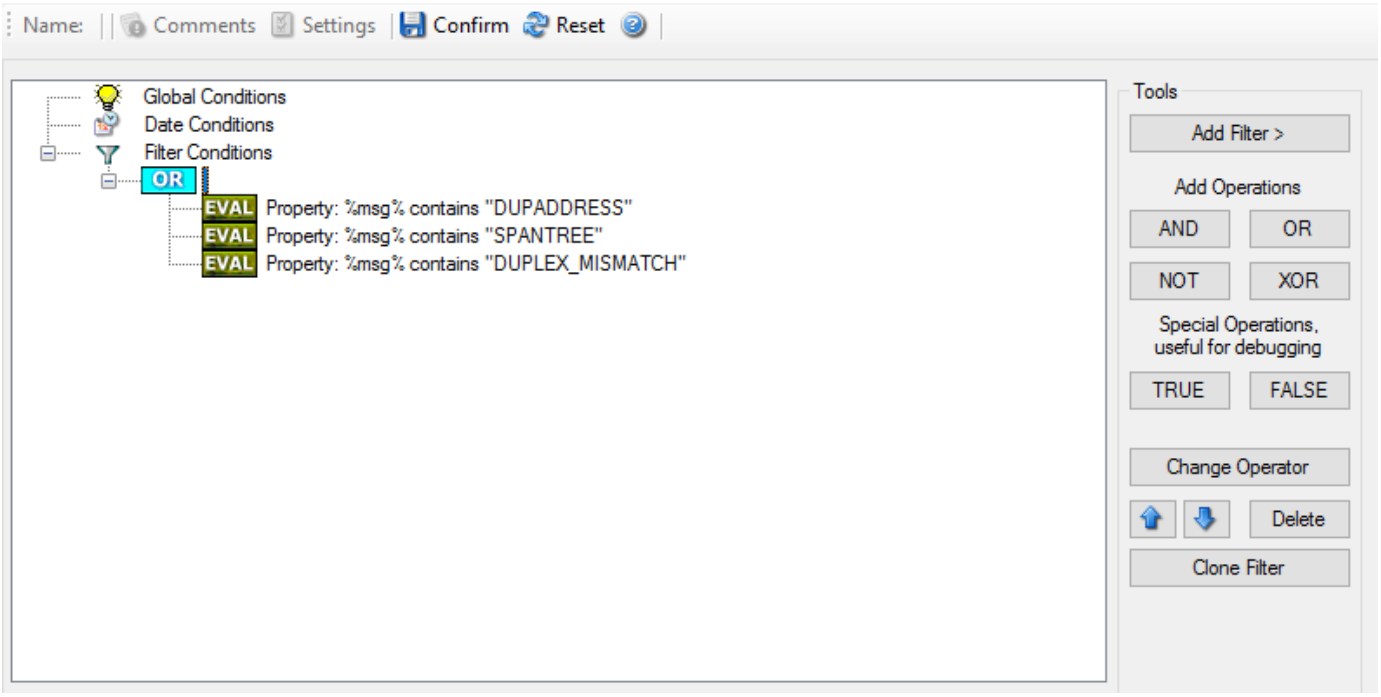


Figure 1 - Example 1

Example 2

Example 2 is very similar to example 1. Again, the message content is to be checked for three string. This time, all of these strings must be present in order for the boolean tree to evaluate to false.

The pseudo code would be as follows (under the same conditions outlined in example 1 above):

```
If (msg  = "DUPADDRES") AND (msg = "SPANTREE") AND (msg = "DUPLEX_MISMATCH) then
    execute action(s)
end if
```

In the filter dialog, this pseudo code looks as follows:

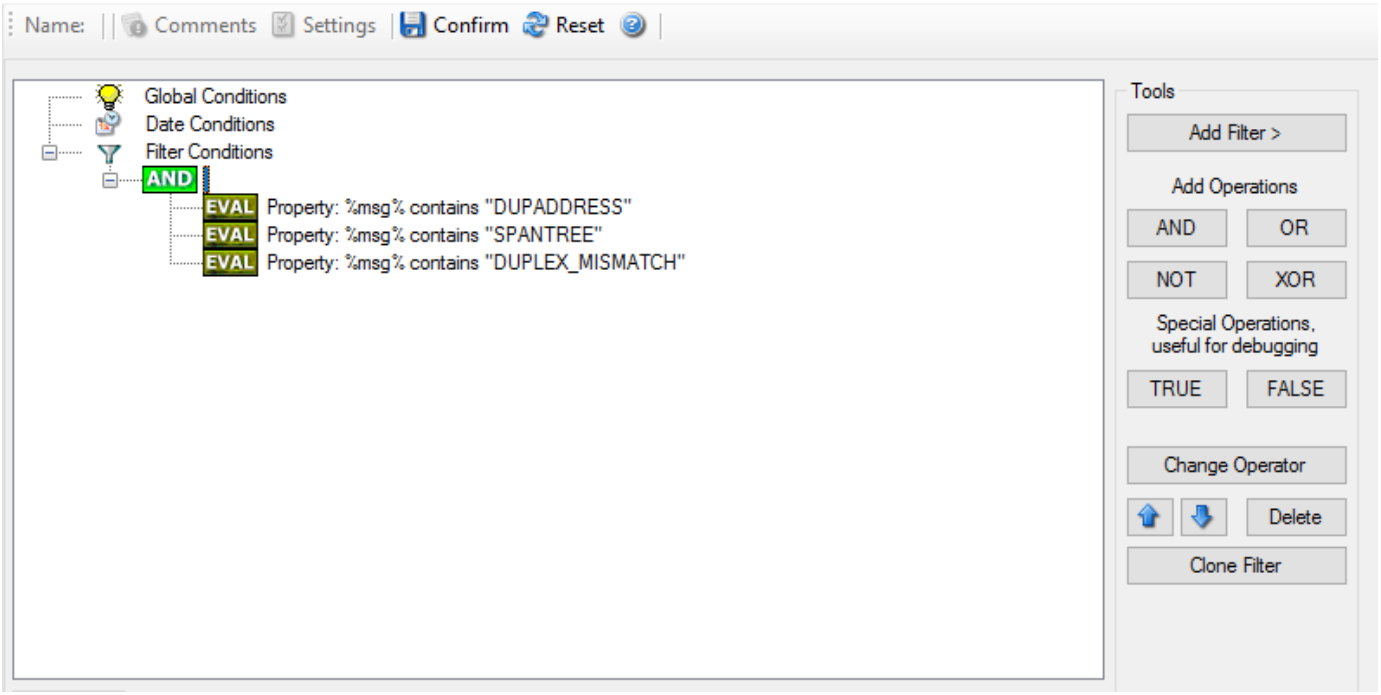


Figure 2 - Example 2

Example 3

This example is a bit more complex version of example 1. Again, the same message text filtering is done, that is if any one of the provided substrings is present, the filter eventually evaluates to true. To do so, the source system must also contain the string “192.0.2”, which can be used to filter on a device from a specific subnet.

An example like this can be used for a rule where the administrator of a specific subnet should be emailed when one of the strings indicate a specific event.

The pseudo code would be as follows (under the same conditions outlined in example 1 above):

```
If ((sourceSys = "192.0.2") And
        ((msg  = "DUPADDRES") OR (msg = "SPANTREE")
            OR (msg = "DUPLEX_MISMATCH))) then
    execute action(s)
end if
```

In the filter dialog, this pseudo code looks as follows:

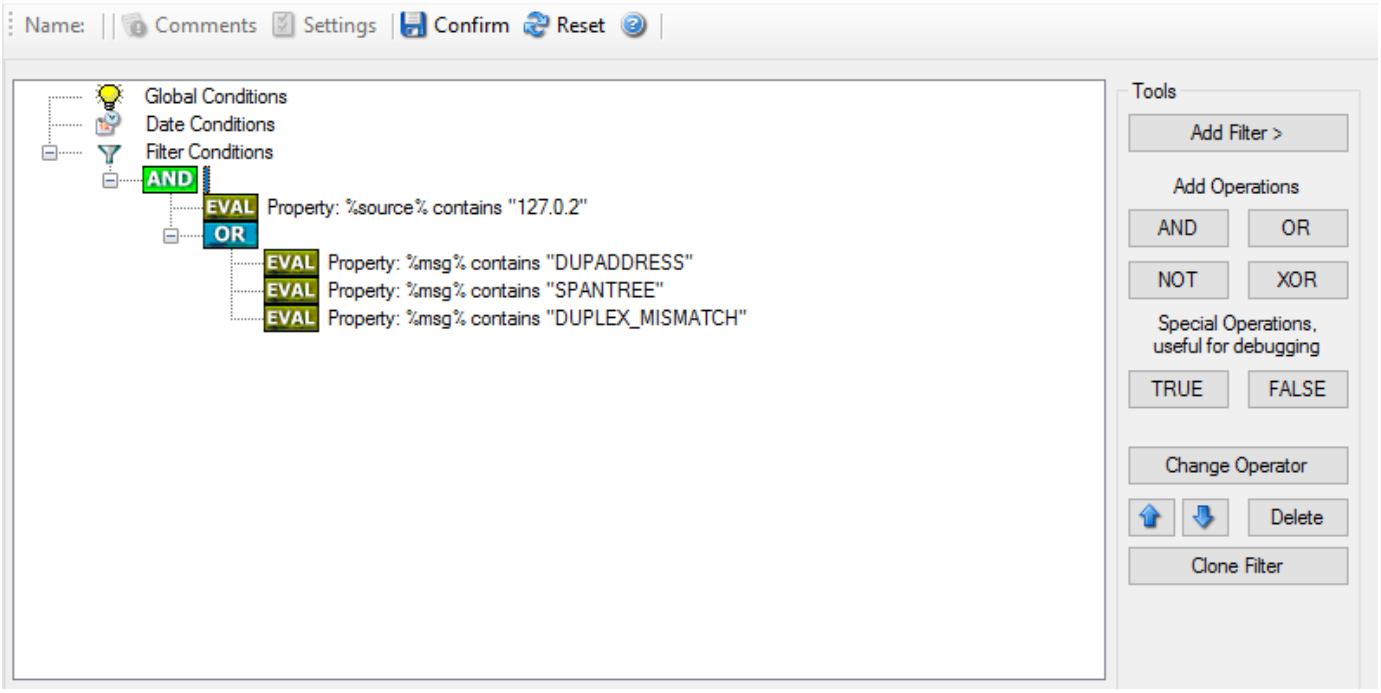


Figure 3 - Example 3

As a side note, you may want to use a range check instead of a simple include for the source system. With a range string check, you can specify that the string must be within a specified column range, in this case obviously at the beginning of the source system IP address.

Real-World Examples

To see some real-world examples of where boolean conditions inside filtering are used, please visit these web links:

- Detecting Password Attacks under Windows

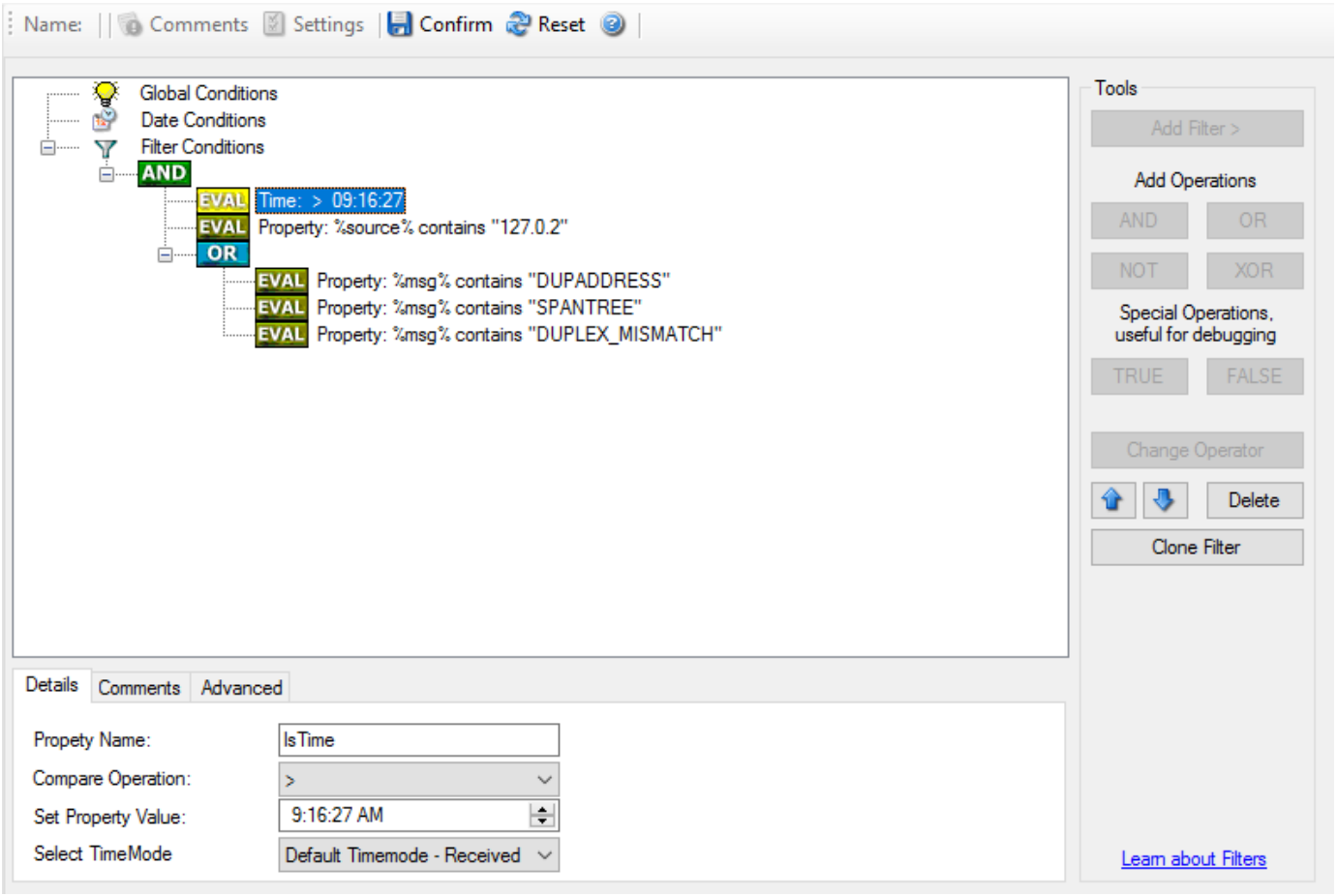
Example 4

In this example, the report is to be filtered in such a way that it shows information only in the case, if the time is greater then certain time with certain event source and one of two event ID's.

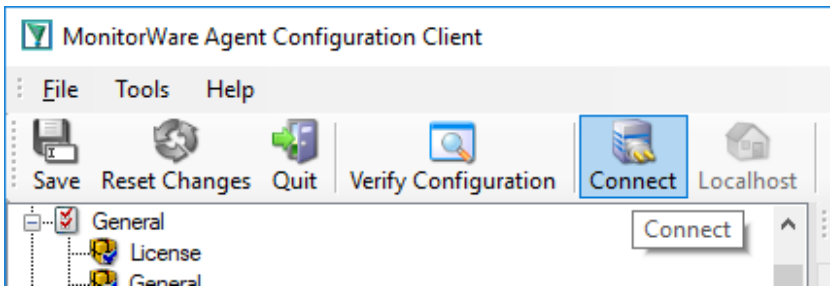
In pseudo-code, the filter could be written like this:

```
If (DeviceReportedTime is greater than {9:16:27} AND EventSource is equal to
{Print} AND [EventID is equal to {10} OR EventID is equal to {18}]]
```

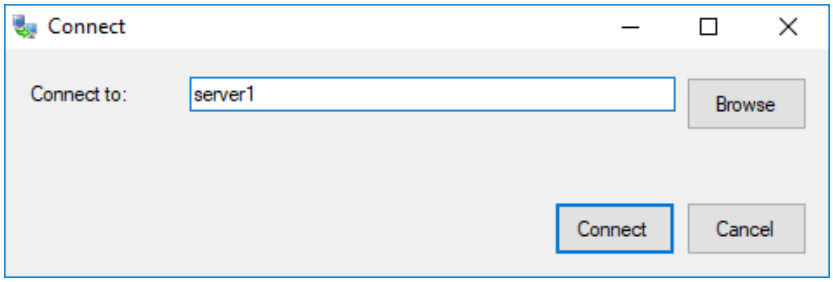
In the filter dialog, this pseudo code looks as follows:



Connect to Computer



Click the Connect button in order to access another machine remotely. A window will open up.



Here you can enter the name of the machine you want to configure remotely. You can either directly enter the name into the textfield or you use the Browse button to see a list of available machines in the network. The click on the Connect button, the configuration client will verify access to the remote machine. If the verification is successful, you will be able to proceed with the remote access. Otherwise an error message will be shown.

Please Note: For remote configurations, you must ensure, that the remote machine

is accessible by network and has access rights for the current logged on local user.

System Error Codes

In most of the cases where you will get system error codes, see this list for their meanings: System Error Codes

If you cannot find them there, please contact us via the Customer Service System.

Glossary

Use this section to look up terms, protocols, and reference concepts that appear throughout the EventReporter manual.

Database

A database is a structural approach to data storage and retrieval. Database systems are optimized for quickly storing and retrieving data. In the MonitorWare products, databases are used to persistently store event data, enabling powerful filtering, searching, and reporting capabilities. MonitorWare supports various database systems including MySQL, MariaDB, Microsoft SQL Server, PostgreSQL, Oracle, and any other database with ODBC support. The database enables you to maintain a centralized repository of all your log data for compliance, forensics, and operational intelligence.

Engine Only Install

An **Engine Only Install** refers to a deployment method where only the core service executable (e.g., `winsyslg.exe`, `mwagent.exe`, `evtlog.exe`) and its essential dependencies are installed, without the full client application and user interface components. This type of installation is particularly useful for:

- **Mass deployments** where you want to minimize the installation footprint
- **Server environments** where GUI components are not needed
- **Automated deployments** where configuration is managed via registry files
- **Security-conscious environments** where you want to reduce the attack surface

In an engine-only install, the service runs with the configuration stored in the Windows Registry, which can be exported from a master installation and imported during deployment. This allows for consistent configuration across multiple systems without requiring the full installation package.

Key characteristics: - Smaller disk footprint (typically just a few MB) - No GUI components or client tools - Configuration via registry import/export - Suitable for headless/server deployments - Can be updated by simply replacing the executable files

This approach is commonly used in enterprise environments where hundreds or thousands of systems need to be monitored with consistent configuration.

IETF

The IETF is an important Internet standards body. IETF is a short name for “Internet Engineering Task Force”. The IETF is responsible for the creation of RFCs. Unlike other, formal standards bodies it is loosely organized. There is no specific membership to the IETF, anyone (knowledgeable) can become an IETF member just by participating on the IETF discussion mailing lists.

The IETF itself provides a good overview over itself at https://www.ietf.org/about/mission/.

IPv6

Adiscon Products officially support IPv6. The IPv6 support was introduced with the following versions:

- MonitorWare Agent 8.0
- WinSyslog 11.0
- EventReporter 12.0

Support for IPv6 is available in all network related facilities of the engine. All network related actions will automatically detect IPv6 and IPv4 target addresses if configured. You can also use DNS resolution to resolve valid IPv6 addresses. Network related Services can either use IPv4 or IPv6 as internet protocol. In order to support both protocols, you will need to create two services. The only exception is the RELP Listener, which uses IPv4 and IPv6 automatically if available.

Registry File

A **Registry File** (with .reg extension) is a text file that contains a snapshot of Windows Registry entries. In the context of Adiscon products (WinSyslog, EventReporter, MonitorWare Agent), registry files are used to export and import complete product configurations.

Registry files enable: - **Configuration backup** - Save your entire product configuration - **Mass deployment** - Apply the same configuration to multiple systems - **Configuration sharing** - Share configurations between team members - **Disaster recovery** - Quickly restore configurations after system failures

The registry file can be created through the product's client interface using the "Export Settings to Registry File" option, and imported silently using: `regedit.exe /s configuration.reg`

This makes registry files an essential tool for enterprise deployments and configuration management.

Resource ID

The Resource ID is an identifier used by the **`adiscon's monitorware line of products`**. It is a simple, administrator assigned string value. It can be used to correlate different events - even from different source - to a specific resource.

For example, on a Windows server running Microsoft Exchange, all Exchange events could be assigned to a resource id of “Exchange Server”.

In MonitorWare Agent and WinSyslog support for Resource IDs is limited. The field is present and can be persisted to the database or stored in XML files, but besides this there is no value in it.

RFC 3164

RFC 3164 is a IETF document. It describes how syslog messages have been seen in traditional implementations. RFC 3164 is not a standard but rather a descriptive (“informational” in IETF terms) document. It does not demand a specific behavior but rather documents what has been seen. Some existing implementations of real-world syslog use different formats.

RFC 3164 is just the first step towards a newer and better syslog standard. A standard already produced by this working group is rfc 3195, which describes how syslog can be sent reliably over a tcp connection.

Adiscon supports RFC 3164 messages. There are a number of switches in each product to take care of those implementation that do it slightly different.

The formal specification for RFC 3164 can be found in the IETF RFC repository.

RFC 3195

RFC 3195 is an IETF standard. It specifies how syslog messages can reliably be transmitted via a tcp connection. RFC 3195 optionally allows for message encryption and authentication of sender and receiver. However, it has not receive any importance in practice. Servers are hard to find.

`adiscon's monitorware line of products` implement the core RFC 3195 protocol (actually, Adiscon was the first one to do this on the Windows platform). Under UNIX rsyslog and SDSC syslog are known to support RFC 3195. Our liblogging project enables your own applications to "talk" 3195.

The formal specification for RFC 3195 can be found in the IETF RFC repository .

During its creation, RFC 3195 was known as "syslog-reliable". Many people still use this name to refer to it.

RFC 5424

RFC 5424 is a IETF document.

This document describes the syslog protocol, which is used to convey event notification messages. This protocol utilizes a layered architecture, which allows the use of any number of transport protocols for transmission of syslog messages. It also provides a message format that allows vendor-specific extensions to be provided in a structured way.

A standard already produced by this working group is rfc 3195, which describes how syslog can be sent reliably over a tcp connection.

Adiscon supports RFC 5424 messages. There are a number of switches in each product to take care of those implementation that do it slightly different.

The formal specification for RFC 5424 can be found in the IETF RFC repository.

SETP

SETP is the “Simple Event Transfer Protocol”. SETP allows reliable delivery of events between SETP supporting systems. EventReporter, WinSyslog, and MonitorWare Agent support SETP. EventReporter works as SETP Client Only. As such, it can forward events generated and gathered by them to central or intermediary SETP servers. WinSyslog EnterPrise Edition works as SETP client and server, only. The MonitorWare Agent can operate both as a SETP server and client and as such also as a relay. It plays a vital role in a complex, distributed environment.

SETP was developed for MonitorWare. It allows synchronous communication between SETP clients and servers. With SETP, an event can be forwarded exactly as it was on the original event generating system. For example, if a syslog message is received on a remote system, that exact syslog message can be forwarded via as many SETP relays as is configured. During that relaying, no information from the original message is altered or lost. As such, each of the relays as well as the final SETP server will see the original source address, time stamps and message.

Furthermore, SETP guarantees reliable delivery. It is based on TCP, so each of the SETP peers know exactly that the communication partner can successfully receive and process the message. SETP guarantees that new events are only forwarded after the previous ones were successfully received and processed. SETP also checks for on the wire errors. Due to its characteristics, SETP can successfully be used in barely or occasionally connected environments like radio connected systems.

The SETP design is influenced by many industry standard movements, most notably the BEEP protocol and XML. However, SETP is optimized to have a very lightweight footprint. As such, it can be implemented even in low powered devices with little overhead.

SMTP

The “Simple Mail Transfer Protocol”. This is an Internet standard for sending email messages. Virtually all major email systems are either based on SMTP or at least offer gateways to SMTP capable systems.

SMTP is used for sending email. It cannot be used to pick up email messages. For this purpose, protocols like POP3 or IMAP4 are required.

SMTP is highly standardized. As such, a standard email client can work with all SMTP compliant servers. In the public Internet, almost all providers offer SMTP compliant mail servers for their customer’s use.

Syslog

Syslog is both a protocol and a system for logging messages in IP networks. Originally developed for Unix systems, it has become the de facto standard for system logging across multiple platforms. Syslog uses UDP port 514 by default (though TCP and TLS variants exist) and follows formats defined in RFC 3164 (traditional) and RFC 5424 (structured). Messages include facility, severity, timestamp, hostname, and message content. All Adiscon products support both sending and receiving syslog messages.

Syslog Facility

Syslog Facility is one information field associated with a syslog message. It is defined by the Syslog protocol. It is meant to provide a very rough clue from what part of a system the message originated from. Traditionally, under UNIX, there are facilities like KERN (the OS kernel itself), LPD (the line printer daemon), and so on. There are also the *LOCAL_0* to *LOCAL_7* facilities, which were traditionally reserved for administrator and application use.

However, with the wide adoption of the syslog protocol, the facility field contents has become a little less clear. Most syslog enabled devices nowadays allow configuring any value as the facility. So it is basically left to distinguish different classes of syslog messages.

The facility can be very helpful to define rules that split messages for example to different log files based on the facility level.

TCP

A reliable IP transport protocol. TCP communication ensures that no packets are lost in transit. As such, it is most useful in low-bandwidth or unreliable environments. Examples are slow WANs or packet radio networks.

Here you find information about Performance Tests and Results

UDP

A non-reliable IP transport protocol. It provides best effort delivery. Typically, in LAN environments UDP packets are never lost. However, in WAN scenarios or with heavily loaded LANs, UDP packets might be lost.

Here you find information about Performance Tests and Results

UTC

UTC is the so-called “universal coordinated time”. UTC was formerly referred to as “GMT” (Greenwich Mean Time) and is the basis of the international time zone system. For example, New York, USA is 5 hours behind UTC. So if it is 12 noon in New York, the UTC time is 5pm.

The MonitorWare line of products often uses UTC. UTC has the fast advantage of providing one consistent time notation, even if devices are across multiple time zones. This is extremely valuable if a central location is to consolidate events from senders in multiple time zones.

Using UTC might not be appropriate if a whole system is contained within a single time zone. As such, most time parameters inside the MonitorWare line of products can be configured to work with local time instead of UTC.

External reference

- Version History

Copyrights

This documentation as well as the actual MonitorWare Agent product is copyrighted by Adiscon GmbH, Germany. To learn more about other Adiscon products, please visit https://www.adiscon.com/en/products.

We acknowledge using these following third party tools. Here are the download links:

Openssl-3.2.1: https://www.openssl.org/source/openssl-3.2.1.tar.gz **Liblogging 0.7.1:** https://github.com/Rsyslog/liblogging/archive/refs/tags/v0.7.1.tar.gz **Librelp 1.11.0:** https://github.com/Rsyslog/librelp/archive/refs/tags/v1.10.0.tar.gz **Libfastjson-0.99.8:** https://github.com/Rsyslog/libfastjson/archive/refs/tags/v0.99.8.tar.gz

Liblognorm 0.3.5 https://github.com/Rsyslog/liblognorm/archive/refs/tags/v0.3.5.tar.gz

Microsoft, Windows, and the Windows logo are trademarks, or registered trademarks of Microsoft Corporation in the United States and/or other countries.

Other mentioned trademarks are for reference only. They belong to their respective owners.

- **genindex**

Index

A

Accessing Properties

C

Command Line Switches
Comparison of Properties
Complex Filter Conditions
Connect to Computer
Customer Properties

D

database
Database Monitor

E

Edition Comparison
engine only install
Event Properties
Event-Specific Properties
EventReporter Shortcut keys

F

Filter Conditions
FromPos

I

IETF
Information Units
Installation
IPv6

O

Options

P

Property

R

Registry File
Resource ID
RFC 3164
RFC 3195
RFC 5424
Rule Engine
Rules

S

Services
SETP
SMTP
Standard Properties
Syslog
Syslog Facility
Syslog Message Properties
System Properties
System Requirements

T

TCP
ToPos

U

UDP
UTC

W

Windows Event Log Properties
Windows Event Log V2 Properties