CorreLog®

Advanced Correlation System User Guide

http://www.correlog.com mailto:info@correlog.com

CorreLog, Advanced Correlation System User Guide

Table of Contents

Section 1: Introduction

This manual provides a guide to the advanced correlation features of the CorreLog Server. The manual provides information on specific features and capabilities of the program related to the higher correlation functions of the system, including operating theory, application notes, and certain features of the system that are intended for advanced users and not documented elsewhere.

The CorreLog Server is easy to get started with, and its basic correlation functions (documented in detail within the User Reference Manual) may be sufficient for most enterprises. However, the CorreLog Server has a number of highly sophisticated features that permit it to perform advanced correlation of messages and data. These are explained in detail herein, including quick reference tables at the end of this manual that document basic correlation rules.

This manual is intended for CorreLog users who require more sophisticated correlation functions than those functions explained in the User Reference Manual. This information may also be of interest to program developers and administrators who want to extend the range of the CorreLog system's role within an enterprise.

For information on other program features, including a more succinct explanation of the correlation functions of the entire CorreLog server, refer instead to the "CorreLog User Reference Manual" and "Correlog Sigma Web Framework Manual", which furnish a complete discussion of all aspects of the system. These reference manuals are provided as embedded documents in all versions of the CorreLog system.

Correlation Function Defined

The word "correlation" has various meanings and different interpretations. The most basic definition is that correlation is simply a relationship between two or more things. The relationship can be parallel, causal, reciprocal, linear, or nonlinear, and be associated with functions of time or other functions. Correlation can be thought of as an expressed function of an independent variable, yielding a dependent value.

CorreLog performs a "semantic" correlation. This semantic correlation is contrasted with purely statistical correlation methods (although statistical functions are provided in various locations within the CorreLog program.)

In terms of expressed function, the input to the CorreLog Server (i.e. the independent variable) is an arbitrary textual message generated by a device, or generated internally by CorreLog. The output of the CorreLog Server (i.e. the dependent variable) is a specific meaning associated with those messages, or in many cases a very specific action that is executed by the program.

Operationally, the CorreLog Server finds "meaning" in the messages, through use of simple or complex match patterns that divide messages into "Threads". Additionally, CorreLog employs "Triggers" to establish context to messages, and "Alerts" to monitor specific message rates. Once the meaning of a message (or group of messages) has been determined, CorreLog takes specific action such as sending a Syslog message, running a program, or opening a ticket and assigning this ticket to a user or group.

The various algorithms and rules, implemented by CorreLog and documented in this guide, provide a huge degree of freedom in establishing "one to N" and "N to one" types of relationships. This correlation process is easy to get started with, but has considerable depth. Because the terms "correlation" and "semantic" are quite abstract, any correlation whose intention is to furnish "meaning" to an arbitrary input stream of data must necessarily have this depth and flexibility, some of which may appear intimidating at first glance, especially without the explanation and application notes furnished herein.

Basic Correlation Components

The CorreLog Server provides four main components. Together, these components can accomplish virtually any correlation required by a user. These components are accessed by the navigation tabs of the web interface, beneath the top-level "Correlation" tab. These components are atomic in nature, that is, they cannot be further subdivided, they do not overlap each other in function, and they can be used as building blocks to create higher and more complex functions. Correlation components are defined as follows:

- **Thread Component.** This is the most basic correlation component of the system, accessed via the "Correlation > Threads" screen of the system. Threads partition raw message data into categories based upon simple or complex match patterns. Each thread consists of a list of received messages that share one or more common aspects, such as they all contain a specific keyword, come from a specific device, and / or occurred during a particular time.

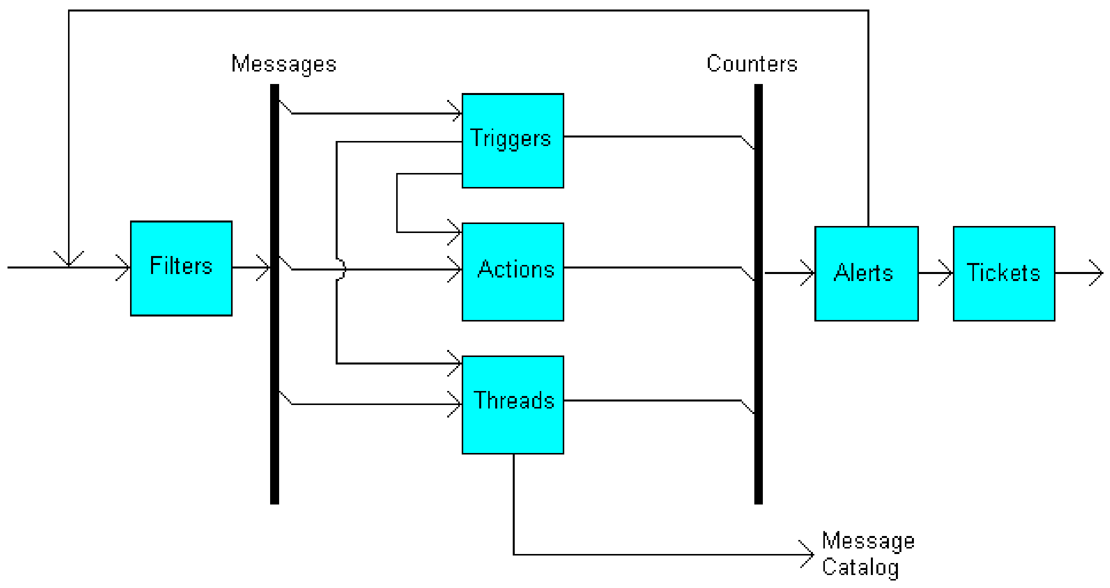
- **Alert Component.** This component, accessed via the "Correlation > Alerts" screen, counts the number of messages received by a Thread and generates a new system message when thresholds are exceeded. The new message is fed back into the main message stream (like any other message) where it can be further correlated. The message is user defined, and describes some special condition, such as too many or too few expected events during a time interval.

- **Trigger Component.** This component is not necessarily required to implement a particular correlation objective, and is often omitted by users as part of their correlation strategy. However, when a Trigger is needed, there is no substitute available. Triggers can be thought of as a message "latches", which retain message information, and enable the gathering of future messages. Triggers are accessed via the "Correlation > Triggers" screen of the system. Each Trigger provides a match pattern, an expiration time, and an optional trigger clear pattern. Triggers are used to establish message context (when needed) such as collecting information when a node starts, or when a specific sequence of messages (such as a data dump) is started.

- **Action Component.** This component, accessed via the "Correlation > Actions" screen, is similar to the Thread component, except this component can take arbitrary action on a message, such as sending notification, updating a database, or opening a ticket on the system. Additionally, Actions can extend CorreLog with high-level correlation functions. Actions can also take automatic action when correlation patterns are discovered. Note that the "Actions" component, while not necessary to achieve any particular type of correlation, can sometimes reduce the complexity of correlation rules via user written programs.

CorreLog Correlation Function Block Diagram

There are various ways of illustrating the operation of the correlation process, depending upon your viewpoint. One simplified block diagram of the CorreLog operation and message dataflow, depicting the relationship of the main components, is shown below.



1. Raw messages from network devices are input into the CorreLog system. These include messages from Unix based devices, network routers, Windows event logs, and application programs.

2. Input messages are optionally filtered and overridden. The user can filter messages based upon content, time of day, device, facility, or any combination of these.

3. Filtered messages are then applied to the various correlation components. Each correlation component maintains a count of received messages.

4. The "Triggers" component provides special application by enabling or disabling threads and actions based upon message content. This permits correlation based on the content of the current message as well as previous messages.

5. The "Threads" component maintains a count of executions, and also list of received messages that have matched simple or complex patterns and specific trigger states. These messages are entered into various message "catalogs", where each catalog is a list of messages corresponding to a specific match pattern.

6. The "Actions" component operates in a fashion similar to the "Threads" component, except can launch external programs based upon simple or complex match patterns and specific trigger states.

7. The "Alerts" facility monitors the "Trigger", "Thread" and "Action" system

counters (as well as other counters) and generates alerts and tickets when thresholds are exceeded.

8. The alerts are fed back into the original stream via standard Syslog protocol for further filtering and correlation. Alert messages are just other Syslog messages distinguished only by the fact that their content is completely supplied and controlled by the CorreLog user.

9. The "Tickets" component is a principle output of the system (as are also the "Message Catalog" maintained by the thread component.) These tickets consist of actionable events.

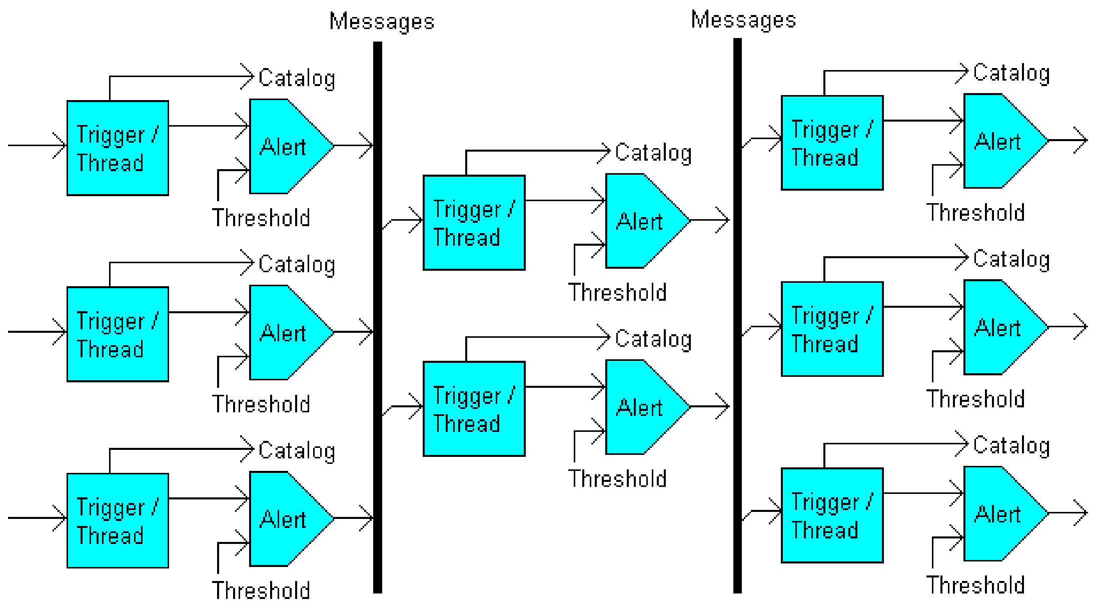
Neural Network Model

It is useful to identify CorreLog as a class of programs called a "neural network". Although it may not be obvious on first inspection, the CorreLog system is actually a neural network, containing "perceptrons" (in the form of Trigger and Alert components) which react to user defined thresholds. Additionally, CorreLog contains summation elements (in the form of Threads) that transfer all input values into a finite and arbitrary set of output values based upon weights assigned to each input.

Neural networks (sometimes referred to as "artificial neural networks") are special software constructs that are common to Artificial Intelligence (AI) software. They operate on the principle that connections between standard components can yield virtually any pattern matching function. They are particularly interesting to researchers and cognitive scientists because they seem to mimic the actual architecture of animal (and human) brains. An outgrowth of these studies has been the widespread opinion that intelligence, and perhaps all mental phenomena, are strictly defined by the types and numbers of connections between simple building blocks.

In practice, neural networks are common programming constructs found in many appliances and software systems. The term "neural network", since its popularization in the 1970's, has become quite common as a means of describing any program that achieves its function through the arbitrary connection of one standard element to another standard element. The concept of neural networks is actually quite old, formulated in the late 19th century as a method of describing the workings of the human nervous system.

The CorreLog system operates as a "recurrent" neural network, which means that all of its output is transferred back into the input of the system, hence is self-aware. The previous block diagram can be redrawn to show how the various correlation components fit the typical neural network model. This diagram is shown below.



1. Input messages are compared to match patterns, and are threaded in cooperation with triggers, creating catalogs of messages. These permit the user to view the reduced messages at any stage of correlation

2. Each trigger and thread combination maintains a count of messages. The counter rates can be detected by the alerting component, which compares the counter rates (over an interval of time) to one or more thresholds.

3. When thresholds are violated, the alert component generates more messages, and these messages can be further correlated with triggers and threads, which can generate further messages, etc.

4. Multiple stages can be created, where messages are "bussed" into all triggers and threads to create a network of correlation rules. The output of each stage is made available as a possible input to all other stages in the system. The above diagram shows three stages, but there can be fewer or more stages within the system.

5. The specific thresholds and connections between each stage of correlation define the types of patterns that are matched. At the final output stage (which can be the first stage, or a much later correlation phase) the highly reduced messages can trigger actions, or open tickets.

One of the characteristics of the above arrangement is that it is well suited for filtering out false positives. This is often cited as one of the most unique and useful aspects of neural network architecture. Neural networks are highly adept and matching patterns in noisy environments, because each stage serves to reduce noise and extraneous false-positives. To increase this filtering action, the user can configure more stages to the neural network.

CorreLog Chatbot Model

Identifying CorreLog as a neural network is valid. However it is also useful to compare the functioning of CorreLog to those of a Chatbot program. This is another type of AI program that has achieved prominence as one of the few techniques for significantly simulating a human interaction. This is a good comparison because, like CorreLog, a typical Chatbot program is also a semantic correlator, which will respond to certain meaningful input.

A Chatbot operates by comparing user input to a series of match patterns, and generating a response based upon the match pattern. The response can contain elements of the original input. Importantly, the match patterns, which trigger a response, can be deeply nested. Therefore, each response depends not only on the current input, but on previous inputs also. This establishes a context for an input based upon one or more previous inputs that have been received.

The CorreLog "Trigger" component provides this capability by permitting a context to be established for particular messages. When a particular input pattern is detected (such as matching a keyword to a message) this sets a Trigger state. The Trigger state can be used to qualify and "gate on" the thread component. Therefore, a Thread collects messages only if a particular previous message has been received.

Each Trigger has an expiration time, after which the trigger state is cleared. For example, the user can set a trigger when a "dump started" message occurs, which enables the "Dump Collection" thread for the next 20 seconds. Triggers can also be cleared by an optional second message, received after the first message. For example, the user can set the trigger with a "dump started" message, and clear the trigger when a "dump finished" message is received.

By default, the "Threads" and "Actions" components permit the user to match the state of a single Trigger as part of the general thread description. As described in later sections, the user can also make Triggers dependent on other triggers, and can make Threads and Actions dependent on multiple trigger states.

Message Keyword Space

Users familiar with Chatbot technology will realize that, although this is an exciting and novel approach to generating artificial intelligence systems, the technology has yet to live up to its promise of passing a "Turing Test". That is to say, it generally quite easy for a person to tell they are communicating with a Chatbot program rather than a real human being.

In short, at this point in time, given the current state of arts in the field of artificial intelligence and computer science technology, Chatbot programs do not really work very well.

What makes CorreLog unique from Chatbot programs, and greatly increases its relevance and usefulness, is that CorreLog works on a much more limited number of inputs than those required by a Chatbot. CorreLog does not have to process all conceivable inputs, but actually operates over a fairly narrow message space.

For example, the word "root" means various things, such as a tree's root, or the result of a math operation, or something that is "core" to an issue. The word "root" is also the universal administrative login name to a Unix platform. In the case of CorreLog, selecting "root" as a match keyword will almost certainly yield only the latter synonym.

The relatively small message keyword space associated with log file messages permits users to configure match keywords for specific cases of log message management, where synonyms are usually limited to just a few special meanings. This is a result of the very nature of log messages, which are typically as precise and brief as possible in order to convey specific meanings to well-versed end users.

In terms of semantic analysis and linguistics, the natural message space for CorreLog has "Cohesive Semantic Similarity", which means that a great many synonymous meanings for a word can be quickly and reliably discarded, increasing the chances that any particular word has an expected meaning.

How To Use This Manual

The information contained in this current section has provided a philosophical basis and description of operation. Subsequent chapters will address the actual application of CorreLog, including a technical description of how to compose and use correlation match patterns, triggers, macros and alerts.

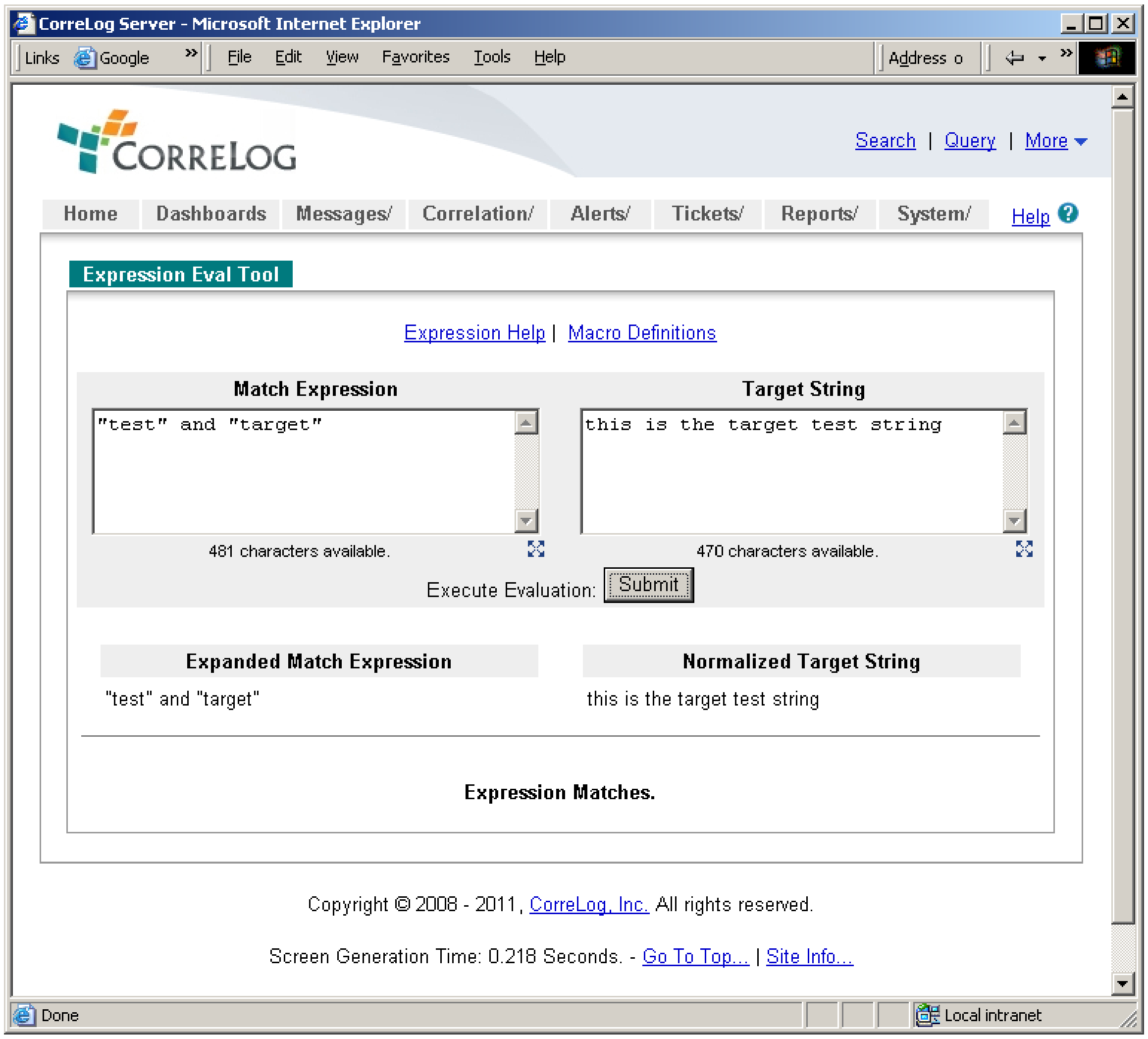
It is highly recommended that the reader log into the CorreLog system and test the various examples illustrated in this manual. This will provide useful practice of how to configure the system, as well as reinforce the concepts stated in this section and the detailed information in the sections that follow.

Note that the companion manual to this one, the "CorreLog User Reference Manual", provides overview of a less technical nature, and contains information on each of these applications, how to navigate CorreLog, and the program's objectives. If you have not yet inspected this companion manual, it is suggested you do so before proceedings with this one.

The Expression Evaluation Tool Screen

As a final note to this first section, it is useful to introduce the user to the "Expression Evaluation Tool" screen. A major portion of this manual deals with a discussion of correlation rules and expressions, presented in the next several chapters. The CorreLog system provides a valuable tool that will assist in understanding the behavior of these expressions.

The "Expression Eval Tool" is accessed via the "More" hyperlink menu at the far right of the display, by the clicking on the "Expr Tool" hyperlink. (No other method of accessing this screen exists, except through the "More" menu.) The screen is depicted below.


CorreLog Server - Microsoft Internet Explorer
Links Google File Edit View Favorites Tools Help Address
CORRELOG
Search | Query | More
Home Dashboards Messages/ Correlation/ Alerts/ Tickets/ Reports/ System/ Help
Expression Eval Tool
Expression Help | Macro Definitions

Match Expression

```
"test" and "target"
```

481 characters available.

Target String

```
this is the target test string
```

470 characters available.

Execute Evaluation: Submit

Expanded Match Expression	Normalized Target String
"test" and "target"	this is the target test string

Expression Matches.

Copyright © 2008 - 2011, CorreLog, Inc. All rights reserved.

Screen Generation Time: 0.218 Seconds. - Go To Top... | Site Info...

As shown above, this tool provides two text input windows: The "Match Expression" window accepts a correlation pattern and the "Target String" window accepts a target string, which is the particular message that the user will test the match expression against.

The user enters a match expression, a target string, and then clicks the submit button to test the expression. The result of the expression evaluation is shown at the bottom of the screen, either "Expression Matches" or "Expression Does Not Match".

Also shown at the bottom of the screen is the "Expanded Match Expression" value (including any global variable and macro substitution, as discussed later) and also the "Normalized Target String (with all letters downcased, and multiple blanks squeezed out of the message.) The "Expanded Match Expression" is particularly useful in learning the behavior of global variables and macros, since the "Expression Eval Tool" shows the final actual match expression, with any macro and global variable names replaced by their actual values.

Using this tool, the reader can quickly test the CorreLog "match expression" capabilities, and learn the general behavior of CorreLog's correlation functions. It is highly recommended that the reader look at this tool now before starting the next section, since the next several sections deal exclusively with a discussion of this capability.

Section 2: Simple Match Expressions

Correlation expressions are the primary building block for the CorreLog engine, and the main way of performing exact data reduction and correlation.

As messages are received by the CorreLog system, each message is compared to all the various correlation expressions configured in the "Threads", "Triggers" and "Actions" screens of the system. When a received message matches a particular expression, an action occurs such as updating the Thread with the message, setting a Trigger state, or executing an Action program. Each message may match multiple correlation expressions configured in assorted locations within the system.

Correlation expressions are simple to get started with, but can also be very specific and lengthy, as discussed here. A correlation expression can consist of a simple keyword, a full word, a keyword wildcard, a quote-delimited string, or a logical conjunction of any these items. Expressions can also reference global variables (which are discussed in the section following this one.)

This section provides a basic overview of simple match expressions consisting of keywords, key phrases, logical operators, and parenthetical nesting. These basic expressions, presented here, are sufficient to satisfy a large number of correlation requirements. More sophisticated expressions, including a description of global variables and comparison operators, are discussed in the sections following this one.

Simple Keyword Matches

Correlation match expressions are text strings, each of which specify a set of received messages that share a common characteristic. Match expressions range from extremely simple matches of keywords to highly involved and lengthy match specifications composed of many different sub-expressions.

The most common type of correlation match expression is a simple keyword, which performs a partial or full-keyword match. The keyword cannot contain any spaces. This type of expression is also one of the most useful. For example, the user can enter the keyword "alert" to match one or more occurrences of "alert", "ALERT", "alerting" or "alerter" anywhere in the body of the received message.

Note that keyword matches are case insensitive. There is no capability (or need) for CorreLog to match a particular letter case. This may initially appear to be somewhat of a limitation, but practice shows that enforcing a policy of case-insensitivity in all matches results in a very high degree of Cohesive Semantic Similarity in received messages (discussed towards the end of Chapter 1).

For example, an expression of the keyword "alert" should naturally match an occurrence of "Alert" in a message, in as much as the words have the exact same meaning apart from their upper or lower case. In becomes clear in practice that a match can always be further qualified by some other mechanism than the letter-case of a particular keyword in the message.

Phrase Matches

By default, keywords in a correlation match expression are delimited by space characters. If a user specifies several keywords within a correlation match expression, all keyword must be present in the target message for the message to match.

For example, specifying a correlation expression of "invalid object access" (without the quote marks) will match the input "the object has invalid access rights", or "access to the specified object is invalid", or "invalid access to the specified object". All three keywords must appear in the input sentence in any order.

It is often the case that that an exact phrase, containing spaces, is required as the match target. To achieve this, the user must enclose the phrase in single or double quote marks to match the entire phrase.

For example, to match a message that precisely contains "invalid object" anywhere in the message, the user should enclose the phrase in balanced single or double quote marks as part of the match expression. This will match both words, and the space, but will not match "invalid" or "object" by itself.

The user can match multiple phrases and keywords in a single message. Each phrase and keyword is delimited by a blank space, and the phrases and keywords will match in any order.

Full Word Matches

By default, a keyword will match either a full word or a keyword. To match a single word, enclose the keyword in quotes with a leading and trailing blank space. For example, the expression " su " (with a leading and trailing blanks) will match "the su user", or "su login", or "login by su" but not "super user successfully logged in."

Note that the leading blank space matches either a blank or the beginning of line, and the trailing blank space matches either a blank or the end of line. This provides an easy mechanism for matching a full word rather than a partial match to part of the message, regardless of whether that word occurs at the beginning, middle, or end of a message.

CorreLog Wildcards

CorreLog implements two wildcards that can further enhance pattern matching. An asterisk "*" matches zero or more characters, a question mark '?' matches a single character. For example "back*up" matches "back up", or "backup" or "file has been backed up." Likewise, the expression "my?file" matches "my file" and "my_file", but not "myfile".

Additionally, one other wildcard is provided, but seldom used: The circumflex "^" character matches the beginning of line. This wildcard is available, but is equivalent to a leading space character in the match expression, as discussed earlier. For example, " su" and "^su" will both match "super user" or "success" occurring at the beginning of the message.

Users familiar with Unix "Regex" type expressions may initially find the set of CorreLog wildcards to be disappointingly limited. However, as shown in later sections, the complexity and obscurity of Regex type expressions does not favor semantic correlation, or "Cohesive Semantic Similarity". Rather, employing a large set of complex wildcards, such as those found in the Regex library, make this semantic correlation highly brittle and tenuous. The smaller selection of wildcards implemented by CorreLog is much better suited to the objective of matching the "meaning" of messages.

If a user finds Regex to be absolutely required, the user can launch a Perl script via the "Actions" program to perform highly targeted pattern matching, as discussed in later sections.

The reader should study other sections of this manual before resorting to this, since other features of CorreLog may make this unnecessary and irrelevant.

Logical Operators

As discussed previously, multiple keywords and phrases can be included in a correlation match expression, delimited by spaces. In this case, to match the expression, the target message must match all the keywords and all the phrases specified, in any order. Each keyword and phrase is joined by an implied logical "and" operator. For example, the correlation match expression "XX YY ZZ" is identical to the correlation expression "XX and YY and ZZ", which matches the message only if all three keywords are present in the message.

CorreLog has a total of four logical operators that can be used to join keywords or phrases in a match expression to build a more complex expression. These operators are well known, and follow the standard rules of Boolean logic.

- **And.** This is the default logical conjunctive operator. For example, "XX and "YY" will match any message that contains both "XX" and "YY", found in any order within the message. Both the left and the right operands to the "and" operator must be present somewhere within the message.

- **Or.** This is the logical "or" operator. For example, the correlation match expression "AA or BB or CC" matches the message if it contains any of the three keywords, found in any order within the message. Either the left or the right operands to the "or" operator, or both operands, must be present somewhere within the message.

- **Xor,** This is the logical "exclusive or" operator, which matches the message if either the left or right operands appear in the message, but not both operands. For example, "QQ xor RR" matches the message "value of qq", and matches the message "value of rr", but does not match the message "value of qq rr" or the message "rrqq exists". The "xor" operator is not used that often, however is invaluable when actually needed.

- **Not**. This is the logical negation operator, which indicates that the keyword or phrase following the operator must not match. For example, "not ZZ" matches any message that does not contain the keyword ZZ. Likewise, the correlation match expression "not AA and not BB and not CC" matches any message that does not contain the all three of the specified keywords, and the correlation match expression "not AA or not BB or not CC" matches any message that contains any of the specified keywords.

Syntactically, these operators are "infix" operators, that is to say they are embedded directly in the expression to affect the expressions meaning. The "and", "or", and "xor" operators require left and right operands. The "not" operator requires a single right operand.

By default, logical operators are evaluated from left to right, which may not be what the user intended. For example, the match pattern "aa and bb or cc" will match the test message "aabb" or "cc", but not "aacc". If an expression has multiple logical operators, it is generally best to use parenthetical nesting to set the order of evaluation, as discussed in the next section.

Parenthetical Nesting

The user can change the order of precedence when evaluating expressions using balanced parentheses. This can also be used to leverage the associative and distributive properties of Boolean logic. By default, if no parentheses are used, then the expression is evaluated from left to right.

Parentheses are often important, and required, when using logical operators described in the previous section. For example, "AA and (BB or CC)" matches any message that contains AA, and also contains either BB or CC or both. This is different from "(AA and BB) or CC", which matches any message that contains both AA and BB in any order, or the keyword CC.

Parenthesis can be nested to any level. When nesting parentheses, the parenthesis must be balanced. For example, "(AA and (not BB)) or (CC and (DD or EE))" is a valid expression, and matches any message that contains AA and not BB, or contains CC and either DD or EE.

CorreLog follows the special rules of the Boolean commutative, distributive and associative laws, as one would expect. The expression "not (AA and BB)" is identical to "(not AA) and (not BB)". Likewise, the expression "(A and B) or (A and C)" is identical to the expression "A and (B or C)". Other Boolean laws, such as the "Identity" and "Redundancy" laws, are also obeyed as one would expect. Good understanding of these laws is not absolutely necessary to creating logical expressions, but is helpful.

Note that, to match the words "and", "or", "xor", "not", or a parenthesis mark as part of an expression, the user simply encloses these special keywords in single or double quotes.

Section Summary, And Additional Notes

1. **Case Insensitivity**. Correlation expression matches are always case-insensitive, without exception.

2. **Simple Keywords.** Correlation expressions can consist of simple keywords, which match any portion of the message. For example "su" matches "success" or "super user".

3. **Phrases**. If the correlation expression contains spaces, it must be quoted. For example "super user" matches "super user" and "super users".

4. **Full Word Matches**. To match a full word instead of a partial match, precede and follow the match expression by a single space. For example " su " matches "the su user logged in", "su login, and "login by su". In the expression, the leading space matches a space or the beginning of line; the trailing space matches a space or the end of line.

5. **Wildcards.** A keyword can contain a "*" wildcard to match zero or more characters, and a "?" wildcard to match a single character. For example "test*fail" matches "the test failed" and "the test did not fail" and also "testfail".

6. **Logical Operators**. To join various expressions into a larger expression, use the "and", "or", "xor" and "not" logical operators. For example "test and not fail" matches any message that contains the keyword "test", and not the keyword "fail", anywhere in the line.

7. **Default Logical And.** If an expression is composed of several sub-expressions without a logical operator, the sub-expressions are joined by an implied "and", and each expression must match the message. For example, while "super user" matches the specified phrase, if the double quotes are omitted, then this is equivalent to "super and user", and the message must contain both keywords in any order.

8. **Parenthetical Nesting.** The user can specify precedence of evaluation using parentheses, which can be deeply nested. For example "(test and file) or (system and user)" matches any message containing both "test" and "file", or any message containing "system" and "user".

Section 3: Advanced Expressions

The previous section described basic correlation match patterns, which form the basic building blocks for "Threads", "Triggers", and "Actions". The user can match a message using keywords, wildcards, and logical operations joining several expressions together, possibly parenthetically nested to change the order of evaluation.

This constitutes a large amount of functionality, but does not address certain common correlation needs such as testing specific fields of the message for certain values, or testing data items that are not part of the message content.

Users familiar with CorreLog will note that the "Thread" and "Actions" screens provide further qualifiers, in addition to the match expression, which permits users to specify match times, addresses, triggers, severities, and facilities. This furnishes a routine method of identify the meaning of messages beyond their message content.

However, this is not the full story. Additionally, all of these special correlation qualifiers can be specified, with complete flexibility, as part of the correlation expression, using specific notation described here. For example, the user can match multiple trigger states, specific word positions, and not contiguous time intervals through specification of special match expressions. These may be applied to any match expression, including trigger expressions.

This section provides a discussion of these advanced expressions, consisting of compare functions and global variables that reference message fields, triggers, and other values.

Message Field References

Previously, the discussion has been limited to specifying keywords and phrases that match any part of the target message. It is also possible to match just a single specific word in the message using message field comparisons. This permits the user to specify what single space delimited word in the message should be tested.

Examples of field specifications include "$3 eq "Test", $4 ne "admin", " su " in $5, and "fail" not in $2. Each of these refers to a particular word in the message (identified by $N) and a comparison function to a constant string.

The "$" character is interpreted by the correlation engine as a global variable, which is immediately substituted in the expression as it is evaluated for some corresponding value. The next section describes a variety of global variables. Here, we introduce the concept of a global variable specifying a particular word in the current message.

Whenever CorreLog sees a "$" character immediately followed by some number within a match expression, it immediately replaces the "$N" value with the "Nth" word in the target message, starting with the first word of the message. If the value of N exceeds the number of words available in the message, the $N reference is not replaced.

This function is mainly useful with "Comparison Operators", discussed in the next section. However, field references also have application when one or two words in the message will match other parts of the message. For example, the pattern "$2.log" will match the message "Test results placed in results.log", because the value of $2 is immediately substituted for "results". (Therefore, this would not match the message "Test results placed in error.log".)

Comparison Operators

The previous section introduced comparison operators, which can be used to compare a particular word in the message with another word. These operators permit expressions such as "(leftvalue operator rightvalue)", returning either true or false.

While the outer parentheses can sometimes be omitted from the comparison expression (depending upon where the comparison actually appears in the expression) it is good practice to include parentheses to promote readability of the expression.

The following operators are available for use with field specifications or global variables.

- **"eq" Equal operator.** The left value must precisely equal the right value, ignoring any letter case. For example, the expression ($4 eq test) returns true if the fourth word of the message is "test" or "TEST", but not "TestCase". Likewise, the expression ($2 eq $5) returns true if the second and fifth words are the same in the message. The comparison is case insensitive and wildcards are treated as ordinary characters.

- **"ne" Not equal operator.** The left value must be different from the right operator, ignoring any letter case. For example, the expression ($4 ne test) returns false if the fourth word of the message is not "test" or Test", but returns true if the fourth word is "TestCase". Likewise, the expression ($2 ne $5) returns false if the second and fifth words are the same in the message. The comparison is case insensitive and wildcards are treated as ordinary characters.

- **"lt" Numeric less than operator.** The left and right values must both be numbers, or begin with numbers. Returns true if the left value is less than the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression ($4 lt 199) returns true if the fourth word of the message is numerically less than 199, or if the fourth word of the message is not a number.

- **"le" Numeric less than or equal to operator.** Similar to the "lt" operator, except the comparison is less than or equal to. The left and right values must both be numbers, or begin with numbers. Returns true if the left value is less than or equal to the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression ($4 le 199) returns true if the fourth word of the message is 199 or less, or if the fourth word of the message is not a number.

- **"gt" Numeric greater than operator.** The left and right values must both be numbers, or begin with numbers. Returns true if the left value is greater than the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression "($4 gt 199)" returns true if the fourth word of the message is numerically greater than 199, and is also a number.

- **"ge" Numeric greater than or equal to operator.** Similar to the "gt" operator, except the comparison is greater than or equal to. The left and right values must both be numbers, or begin with numbers. Returns true if the left value is greater than or equal to the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression "($4 ge 199)" returns true if the fourth word of the message is 199 or greater, and is also a number.

- **"llt" Lexical less than operator.** This performs an alphabetical comparison of the left and right values. Returns true if the left value is alphabetically less than the right value, ignoring case. For example, the expression "($1 llt T)" returns true if the fourth word of the message is "success" or "Success", and returns false if the fourth word is "testcase", "Test", or "workgroup"

- **"lle" Lexical less than or equal to operator.** Similar to the "llt" operator, except the comparison is alphabetically less than or equal to. Returns true if the left value is alphabetically less than or equal the right value, ignoring case. For example, the expression "($1 lle T)" returns true if the fourth word of the message is "success" or "TEST", and returns false if the fourth word is "workgroup" or "Workgroup".

- **"lgt" Lexical greater than operator.** This performs an alphabetical comparison of the left and right values. Returns true if the left value is alphabetically greater than the right value, ignoring case. For example, the expression "($1 lgt T)" returns true if the fourth word of the message is "workgroup" or "Workgroup", and returns false if the fourth word is "testcase" or "success".

- **"lge" Lexical greater than or equal to operator.** Similar to the "lgt" operator, except the comparison is alphabetically greater than or equal to. Returns true if the left value is alphabetically greater than or equal the right value, ignoring case. For example, the expression "($1 lgt T)" returns true if the fourth word of the message is "test" or "TEST" or "Workgroup", and returns false if the fourth word is "success" or "Success".

- **"in" Keyword or wildcard match pattern.** This operator tests to see if the keyword or wildcard contained in the left value is found in the right value. This is similar to the correlation expressions discussed previously, but is confined to the value contained in the right value. For example, the expression "(key" in $5)" returns true if the fifth word of the message is "key", "keyword" or "PASSKEY". Likewise, the left value can contain wildcards. For example the expression "(key*o*d in $3)" matches the third word if it is "Keyword" or "keyboard".

- **"not in" Keyword or wildcard match pattern negation.** The same as the "in" operator, except that the comparison returns true if the left value keyword or wildcard is not in the right value. . For example, the expression "(key not in $5)" returns false if the fifth word of the message is "key", "keyword" or "PASSKEY", and the expression "(key*o*d not in $3)" returns false if the third word if it is "Keyword" or "keyboard". Note that this is the only comparison function containing two separate words.

Comparison functions permit the user to target very specific portions of the target message. As with other expressions, comparisons can be joined together using logical operators to create very specific match patterns. For example, a typical expression might be ($1 eq "Test") and (("Key" in $5) or not "Failure"). This pattern matches a message where the first word of the message is "test", and either the fifth word of the message contains the string key, or the word "failure" does not appear anywhere in the message.

Message Related Global Variables

In the previous section, the concept of a global variable was introduced by explaining "Field References". Whenever the CorreLog system finds a word in an expression consisting of a dollar sign "$" followed by a number, the symbol is immediately replaced by the corresponding word in the message. This occurs prior to any evaluation of the expression.

The "$" character identifies a global variable that is maintained by the CorreLog system. In addition to field references, such as "$1, $2, $3", etc., there are a number of other global variables that can be used in expressions. These global variables are set each time any message is received. They can be used as keywords (matching any part of the message) or used in comparison functions. The list of global variables is as follows:

- **"$address" Global Variable.** The word "$address", when found in a correlation expression, is immediately replaced by the IP address of the device that sent the message. For example, if the user includes "$address" as a keyword in the expression, the IP address of the device that sent the message must appear somewhere in the message. More commonly, this value is used in a comparison function such as ("10.1.2.2" eq $address), which matches only if the device that sent the message has an address of "10.1.2.2". This provides an alternative to specifying an address in the "Match Address" specification of thread and action screens.

- **"$username" Global Variable.** The word "$username", when found in a correlation expression, is immediately replaced by the username contained in the message (if any) or the string "none" if the message did not contain a username. This value can be used in a comparison function such as ($username eq "jsmith"), which matches only if the keyword "jsmith" is in the message, and "jsmith" is also a username defined in the "Messages > Users" screen.

- **"$userdata" Global Variable.** The word "$userdata", when found in a correlation expression, is immediately replaced by the user group information residing in the "./config/userdata.cnf" file corresponding to the specified username. The username for the message (if any) is used to index the user data. This permits correlation on user information that is external to the message, such as the user's full name, location, e-mail address, or any other data in this configuration file. For example, the value can be used in a comparison function such as ("New York" in $userdata), which matches if the username, specified in a message, has "New York" as part of the $userdata record.

- **"$devname" Global Variable.** The word "$devname", when found in a correlation expression, is immediately replaced by the device name associated with the IP address (if any). The device name is the value configured in the "Device Information" screen by the operator, which may or may not be the official DNS name for the IP address. For example, the value $devname eq "localhost" matches only if the IP address of the device that sent the message has a name "localhost" defined in the "Device Information" screen.

- **"$devdata" Global Variable.** The word "$devdata", when found in a correlation expression, is immediately replaced by the device group information residing in the "./config/devdata.cnf" file, corresponding to the specified address. The address of the message is used to index the device data. This permits correlation on device information that is external to the message, such as the purpose of the device, its location, its asset information, or other information. For example, the value can be used in a comparison function such as ("Solaris" in $devdata), which matches if the IP address, specified in a message, has "Solaris" as part of the $devdata record.

- **"$facility" Global Variable.** The word "$facility", when found in a correlation expression, is immediately replaced by the facility of the current message. This will be either the official facility name, such as "kernel", "user", "network", or can be a user-defined facility. (User defined facilities are discussed in the "CorreLog User Reference Manual", and are configured under the "Messages" navigation tab of the system.) This global variable provides an alternative to specifying a facility in the "Match Facility" specification of thread and action screens, and additionally permits a range of facilities to be specified as part of a match pattern.

- **"$severity" Global Variable.** The word "$severity", when found in a correlation expression, is replaced by the severity name of the current message. This will be the official severity name, ranging from "debug" to "emergency". This global variable provides an alternative to specifying a severity in the "Match Severity" specification of thread and action screens. This value, while useful, is generally not as powerful as the "$sevnum" global variable, which is the corresponding numeric value of the severity, and which permits the "lt", "le", "gt", and "ge" numeric comparisons to be made on a severity value.

- **"$facnum" Global Variable**. The word "$facnum", when found in a correlation expression, is replaced by the facility number of the current message. This will be the numeric value of the message facility, ranging from "0=Kernel" to "24=Other". (User defined facilities, which by their nature have no numeric value, are assigned a number of 24, following the last official facility number of 23=Local7). This value permits the "lt", "le", "gt", and "ge" numeric comparisons to be made on a facility value.

- **"$sevnum" Global Variable.** The word "$sevnum", when found in a correlation expression, is replaced by the severity number of the current message. This will be the numeric value of the message severity, ranging from "0=emergency" to "7=debug". (Note that the Syslog protocol defines the highest severity to be zero). This value permits the "lt", "le", "gt", and "ge" numeric comparisons to be made on the received message severity value. For example, this permits an expression such as "($sevnum ge 3) and ($sevnum le 6) " which matches any message with severities of "info", "notice", "warning" and "error".

- **"$date" Global Variable.** The word "$date", when found in a correlation expression, is replaced by the current date, in "YYYY/MM/DD" format. This may be useful when configuring correlation patterns specific for particular days and / or months, especially when used with the "in" and "not in" comparison operators. For example, the expression ("/??/01" not in $date) will match a message only if it is not the first day of the month.

- **"$wday" Global Variable.** The word "$wday", when found in a correlation expression, is replaced by the current three letter weekday abbreviation, either "mon", "tue", "wed", "thu", "fri", "sat", or "sun". This may be useful when configuring correlation patterns specific for particular days of the week. For example, the expression "($wday ne tue)" will match a message only if it is not received on a Tuesday.

- **"$time"Global Variable.** The word "$time", when found in a correlation expression, is replaced by the current time, in "HH:MM:SS" 24 hour format. This may be useful when configuring correlation patterns that match specific times, extending the "Match Time" specification of thread and action screens. This global variable is typically used with a lexical compare function, such as "llt" or "lgt".

Other Global Variables

In addition to the global variables described in the previous sections, there exists a whole class of global variables related to CorreLog "Triggers". This permits the user to specify, not only the $address, $severity, $facility, etc., of the current message, but these values found in previously received messages.

This greatly expands the capabilities of the system to work with message "contexts", and target situations such as the current message matching a particular keyword found in a previous message. Because of the specialized nature of these messages, and the use of CorreLog Triggers in general, this discussion is provided separately, in the next section.

Unmatched Or Non-existent Global Variables

Note that if the user specifies a global variable in an expression that is not set, or misspells the global variable, the variable is not replaced. The variable is simply regarded as another word in the expression and CorreLog tries to match the word in the message. For example, if the expression is "$addr eq 10.1.1.1", then this will always fail, because the user misspelled $addr (instead of $address, the proper name for the global variable.) Since the word "$addr" is not equal to the word "10.1.1.1", this always returns false. The user should therefore be careful when using any of the global variables to avoid unanticipated behavior.

Expression Evaluation Tool

As a final note to this section, the behavior of both basic and advanced expressions, including all elements described in this section, may be observed using the "Expression Evaluation Tool" introduced at the end of Chapter 1. In particular, this tool will expand most (but not all) of the global variables listed above, substituting the global variable name for its value. The results of the expansion can be seen at the bottom the screen after clicking the "Submit" button.

This tool, accessed via the "More" menu hyperlink, allows a user to test and experiment with keywords, wildcards, compare functions, and global variables. The reader is encouraged to access this tool, and apply any of the examples discussed so far.

Section Summary, And Additional Notes

1. **Global Variables.** CorreLog establishes various global variables that can be used in expressions. Global variables are all preceded with a "$" dollar sign and are substituted in the expression before it is evaluated.

2. **Word Position References.** The user can use the $N global variable to refer to a particular word in the message, where $1 is the first word of the message, $2 is the second word of the message, etc.

3. **Compare Functions.** The user can specify compare functions, either "eq", "ne", "lt", "gt", "le", "ge", "llt", "lgt", "lle", "lge", "in" and / or "not in" as part of the expression. These can be used to match global variables and

word position references. For example, ($3 eq test) matches if the third word in the message is "test". Also, (test in $8) matches the message if the eighth word in the message is "test" or "testfile" or "runtests".

4. **Mixing Compare Functions And Keywords.** Compare functions can be mixed with other expressions, such as keywords or wildcards, and can be parenthetically nested. For example, the moderately complex expression "($2 eq username) and ($3 eq smith) and logon" might indicate that username "smith" has logged into the system.

5. **Other Global Variables.** In addition to word position references in the form of $1, $2, $3, there are other global variables such as $address, $severity, $time that can be used in expressions, especially useful with compare functions. These are set each time a message is received. For example ($address eq 10.1.2.3) is set if the IP address for the received message equals "10.1.2.3".

6. **Unmatched Global Variables.** If a global variable is not set or is misspelled, the global variable is regarded as a keyword and is not substituted in the expression.

Section 4: Correlation Alerts

The "Alerting" component of the system has been mentioned previously, and is central to the discussion of CorreLog and correlation applications. This component, configured in the "Correlation > Alerts" tab of the system, accepts as input a system counter, a counter threshold, and a test interval.

The Alert facility works with a wide selection of internal CorreLog system counters, allowing the user to detect when system counters change, or exceed certain rates of change. The "Thread" counters are commonly used as the target of alerts, however other counters (such as username, device, facility, and severity counters) can also be monitored, expanding the role of Alerts on the system to include many types of system data.

The alerting component has two distinct outputs: a Syslog message and an optional "Ticket" which is assigned to a user or group.

A Syslog message is always sent by the Alerting component whenever a threshold is violated. This message is fed back into the list of messages for further correlation. The alert message, which is completely under the control of the user, describes the over-limit situation and can be used to run an action program, or can be used to increment more counters which are further alarmed.

A "Ticket" can be optionally opened for any system user, or for a "Ticket Group" whenever a threshold is violated. This provides a way of monitoring and dispatching alerts beyond simple logging of alert conditions, and can incorporate workflow features and incident management policies.

Creating A New Thread And Alert Combination

The most common counters used with the Alerting component are the "Thread" counters, which tabulate the number of messages received for a particular thread. Therefore, the user creates a thread, and then creates an alert on the counter threshold. A typical configuration scenario is as follows.

1. The user creates a Thread, in the "Correlation > Threads" screen of the system, which matches a particular group of messages. This can be a simple match pattern, or more complex match pattern that uses triggers.

2. The user creates an Alert in the "Correlation > Alerts" screen of the system, which places a threshold on the thread counter, created above. (More information on these thresholds is provided in the next section of this manual, but a typical value of 3 counts per 60 seconds is often a good starting point. The "Wizard" button of the "Add New Alert" screen is of assistance in configuring this alert.)

3. The user configures an Action program, in the "Correlation > Actions" screen, which looks for a precise message or keyword contained in the Alert message, configured above. When the thread logs a certain number of counts, the action program is executed.

For example, by following the above procedure, it is possible to have an e-mail message sent when the number of invalid logins during a 60 second period exceeds three invalid logins. This is trivial, but typical application of CorreLog.

Note that there are other system counters that can be monitored. The list of counters is available via a drop down menu on the Add New or Modify Alert screen, and include all threads, all triggers, and also facility and severity catalogs, as well as certain global system counters.

Setting Alert Thresholds

As can be seen from above, the alert threshold is fairly important. If the user selects a threshold that is too high, then the action program is not executed in a timely fashion. Likewise, if the threshold is too low, then the action program is executed too often, creating false positives.

In practice, it is not as difficult to set alert thresholds as one might think it is. As suggested earlier, a threshold of 3 counts per 60 seconds is often the exact setting required to generate meaningful alerts, especially when a message occurs relatively infrequently, or a message is sporadically received

This value of "3" actually has a solid mathematical basis as follows: If the standard deviation of a data set over a relatively small interval is less than 1, and the average of that data set is close to zero, then the probability distribution of the data is best given by the "Gauss Error Function (also denoted "erf(x)"). The probability of 3 messages occurring over that interval is: 1 - erf(3 / sqrt(2)), which evaluates to approximately 0.5% of all the sampled time intervals. Therefore, for this type of typical data, and with a sample interval of 60 seconds, the typical alert will be triggered approximately once every three hours or less.

If both the average value and standard deviation for the sampled data are greater than 1, then a regular Gaussian normal distribution provides a more appropriate estimate of an alert occurring. In this case, the user can select three standard deviations away from the average for a meaningful alert, which again evaluates to approximately 0.5% of all sampled time intervals.

Alert Threshold Hints

These alert values are automatically calculated for a user via the "Threshold Hints" screen, accessed via a hyperlink on the Add New or Modify Alerts screen. This screen provides recommended settings for an alert, given a specific time interval. In most situations, the user can accept this as a baseline and then make adjustments if needed during the course of CorreLog operation.

The alert threshold hints screen displays thresholds for recommended severities. (In general, severity of an alert is something that is very user specific, and it cannot be easily guessed programmatically.) The thresholds increase based upon the severity, where "critical" events are assumed to occur much less frequently than "info" and "warning" events. This is a good assumption, since the user necessarily will be attempting to reduce criticality of messages as part of the normal operation management of the enterprise, hence critical events will in fact occur less frequently.

The "Threshold Hints" are available on the "Alert Edit" screen, and also can be viewed for any catalog of messages by clicking the "View Catalog Statistics" hyperlink towards the bottom of these screens.

Auto-Learn Function

The user can drill down into each alert, and set the threshold recommended by the "Alert Hints" screen. However, this function is actually performed automatically by using the CorreLog "Auto-Learn" function (accessed via the "Correlation > Config > Auto-Learn" tab.) This greatly simplifies the setup of the system, especially after initial installation.

By default, CorreLog monitors message rates for ten days after installation (or after a new thread is added) and derives message statistics each night, shortly after midnight. Given proper conditions (such as the proper amount of data for a thread) the alert thresholds are automatically adjusted based upon the severity of the alert, the average message count, and the standard deviation from the message average. Additionally, the Auto-Learn function can look at the number of tickets that have been opened by the alert, and increase the alert threshold so as to reduce the number of tickets.

The Auto-Learn function allows the user, when he or she creates a new alert, to use a default value for the alert threshold. Over the course of time, the system will monitor the counters for the alert and set the threshold for the alert to an appropriate value.

Alert Formulas

Alert Formulas are an advanced feature of the alerting facility. This feature permits a user to set thresholds on any system counter, including those that do not explicitly appear in the "System Counter" drop down menu on the "Alert Edit" screens. In particular, users can set thresholds on device address message counters, even though these are not explicitly listed in the drop down menus.

Additionally, the user can create complex formulas that reference multiple counters, such as summing up various counters with specific weights. For example, a user can set a threshold on the sum of various counters, where each counter is first multiplied by a specific factor. Or, the user can set a threshold on the "log10" value of a counter, to compress the counter value into a logarithmic scale.

The "Alert Formulas" screen, access via the Correlation> Config navigation tab, furnishes this capability. This screen provides special utility in creating special math formulas that work with the CorreLog "Alerts" facility. The user first creates a formula, using special notation to reference one or more system counters. Once a formula is created, it appears in "Alerts" component edit screen (with a "Formula/" prefix) and can be subsequently selected like any other system counter.

Referencing System Counters In Alert Formulas

Each "Alert Formula" consists of a user selected name and value. The value contains references to any of the various counters on the system. Each counter is referenced using the notation "$(type)/(uid)", where the type is either "address", "thread", "trigger", "facility", "severity", or "system". The "uid" portion of the reference depends upon the type, as follows:

- **$address/(ipaddr).** The user can reference any device counter by specifying the keyword "$address/" and appending the IP address of the device. For example, the value "$address/10.1.2.1" represents the number of messages received for the device with IP address 10.1.2.1. If the 10.1.2.1 device does not exist, the value substituted in the formula is zero.

- **$facility/(facname).** The user can reference any facility counter by specifying the keyword "$facility/" and appending the official name of the facility. For example, the value "$facility/kernel" represents the number of messages received with a facility of "kernel". The user can specify user defined facilities, or one of the standard facilities. If the facility does not exist, the value substituted in the formula is zero.

- **$severity/(sevname).** The user can reference any severity counter by specifying the keyword "$severity/" and appending the official name of the facility. For example, the value $severity.info" represents the number of messages received with a severity of "info". If the severity does not exist, the value substituted in the formula is zero.

- **$trigger/(trigname).** The user can reference any trigger counter by specifying the keyword "$trigger/" and appending the name of the trigger. For example, the value "$trigger/coldstart" represents the number of match counts for the "ColdStart" trigger. If the ColdStart trigger is not found, the value substituted in the formula is zero.

- **$thread/(uid).** The user can reference any thread counter by specifying the keyword "$thread/" and appending the "Thread > UID" for the thread. Note that this is the unique identifier for the thread (not the thread name), available only from the "Audit" hyperlink for the "Correlation Threads" screen. (This will be a 12 or more digit integer number, possibly with leading zeros. The value identifies the thread uniquely, and never changes.) For example, the value of "$thread.000000114915" corresponds to the thread with the unique identifier 000000114915 on the system, as shown by the "Audit" hyperlink at the bottom of the "Correlation > Threads" screen.

- **$system/(ctrname).** In addition to the above counters, the user can reference one of the system counters, either system/messages (which is a count of all messages received) or "system/actions" (which is a count of all actions launched) or "system/triggers" (which is a count of all triggers matched.)

Note that it is irrelevant as to whether the above values reference the History or the Current counters since the Alert facility always operates on the difference (delta) values of the counters. Further note that all the above references are case insensitive.

Evaluating Alert Formula Math Expressions

What gives the Alert Formula facility its special power is that the user can apply math expressions to each of the above references using standard "+" to add, "-"

to subtract, '*" to multiply, and "/" to divide. The user can reference multiple counter values, or scale any counter value. For example, the user can create a formula named "Critical_Or_Higher" that has a value of "$severity/critical + $severity/alert + $severity/emergency", and this formula yields the sum of all of these counter values. After creating this formula, the value "Formula / Critical_Or_Higher" is configured as the system counter name in the "Alerts" screen.

Math expressions can be parenthetically nested to change the order of evaluation (which is normally a simple left to right precedence). Additionally, the user has various math functions available, each of which can also be nested. These math expressions include all the functions of the standard C language math library, such as "sqrt ()", and "log()" and "log10()" functions.

The result of any math expression is always an integer number, rounded down. The value is clipped at either zero as a minimum or 50 counts as a maximum. (Dividing by zero yields a value of zero.) This is because each alert stores a maximum of 50 related messages; hence this is the range of thresholds that are acceptable as input values. If the user is looking for more than 50 counts per minute, then the sample interval should be decreased.

Alert Configuration Policies

In practice, the user sets the threshold for the Alert by either entering a numeric value of 3 counts per 60 seconds to get started with, or by drilling down to view the "Threshold Hints" for the alert counter and interval, based upon any data that has been previously collected. This greatly simplifies the otherwise potentially difficult task of setting these thresholds. If the alert is triggered more than desired, the interval can be expanded and / or the threshold can be increased to reduce the number of alerts.

The alert thresholds are related to the severities quite simply. If someone adjusts a threshold upward, and makes the alert less likely to occur, the severity of the alert should probably be increased. Likewise, if the test interval is increased, this causes the alert to be averaged over more time, decreasing the likelihood of the alert occurring. Therefore, if the user increases the test interval for an alert, the user should also consider decreasing the alert severity.

Alerts are great candidates for the "User Defined Facility" function of CorreLog discussed earlier. Since the user has complete control over the textual content of the Alert, the user can easily setup keywords that cause the Syslog message to appear in the Messages with a particular user defined severity. This makes it easy to identify the Syslog messages generated by alerts, and also makes the further correlation of these messages easier. For example, the user can incorporate the keyword "MS-Exchange" in the alert message. The user can then create a facility override (as discussed previously) that sets this message with an

"MS-Exchange" facility code.

As is the case with Threads, there can be a multiplicity of Alerts. If an Alert is seldom used, it is harmless, and quickly drops to the bottom of the list. Not every Thread will require an alert. (Some Threads make exist strictly for information, and do not need special alerting.) Likewise, a thread can have multiple alerts at different thresholds. The user can audit the full list of Alerts via the "Audit" hyperlink at the bottom of the display.

Section Summary, And Additional Notes

1. Alerts operate on counter rates for system counters. These counters can be user defined threads, as well as a variety of other counters on the system.

2. When an alert is set, it sends a syslog message back to CorreLog with a user-defined message. This message can then be further correlated by the system.

3. When an alert is set, it can optionally open a ticket and assign it to a system user. This keeps track of significant events on the system, and creates actionable data for operations.

4. Users can see the recommended thresholds for alerts by clicking the "Alert Threshold Hints" link on the "Alert Edit" screen. These thresholds hints are derived from message rates, and standard deviation intervals from the message rate average.

5. The recommended thresholds for alerts can be automatically applied to alerts using the CorreLog "Auto-Learn" function, which adjusts alert thresholds up or down each night, shortly after midnight, for a specified defined period of days (by default 10 days after installation or creation of a new alert.)

6. Alert Formulas are user-defined combinations of counters that are combined together via math equations. This permits multiple counters to be combined into a single counter, possibly weighting and scaling each counter as part of the formula.

7. The user can define specific facility codes associated with alerts, which assist with correlation. This is especially easy because the user can incorporate unique keywords or phrases into the alert text.

Section 5: Correlation Triggers

The previous section discussed how to correlate messages as they arrive at the CorreLog system. Because error messages, by their nature, do not usually need any special context to be understood, the prior sections have ignored message context, and still managed to describe a wide range of useful correlation functions.

A log message, by itself, is typically a stand-alone communication; hence it is not usually necessary to worry about the nature of any earlier message. However, there are many notable times when the full meaning of some message depends upon some other, different message that has been previously received.

For example, an error message that has been preceded within the last several minutes by a "reboot" message will signify something slightly atypical (that is, a startup error, as opposed to some other type of error.)

The purpose of the CorreLog "Triggers" function is to provide the additional functionality needed to place a message within a specific context. As discussed in this section, CorreLog "Triggers" furnish a special type of "time" correlation, which takes into account not only the current message, but also previous messages that have been received.

Human communications is highly dependent upon context to derive meaning. Chatbot programs recognize this, and construct state machines to handle context. Similarly CorreLog provides Triggers to establish a state machine, and a high-degree of semantic correlation, as discussed herein.

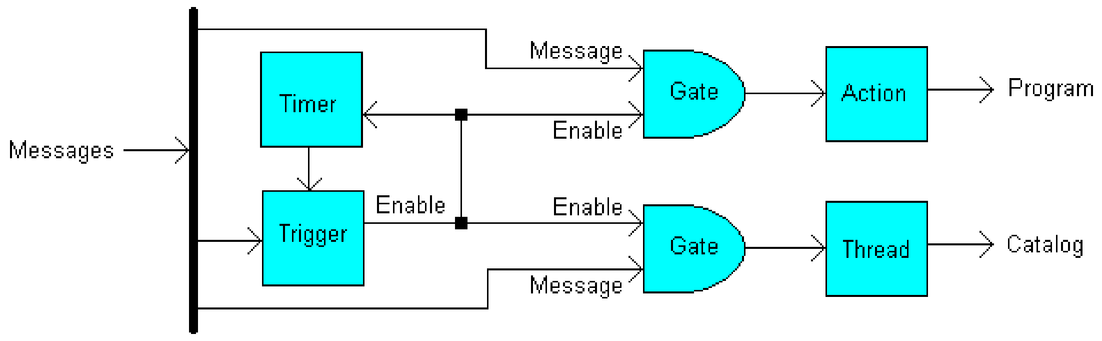
Trigger Function Defined

Triggers are explicitly referenced in the "Match Trigger State" fields of the "Threads" and "Actions" screen, where they select messages in a fashion identical to the other message qualifiers, such as the match expressions discussed extensively in the previous sections of this guide.

When an Action or a Thread references a trigger, that trigger further qualifies the received message. Specifically, in addition to that message matching all the criteria of the Thread or Action (such as a time range, IP address, facility, severity, and match expression) the specified trigger state (either "set" or "cleared") must also be matched.

The trigger "gates" the processing of messages by the particular Thread or Action component. Messages are selectively processed based upon some previous message that has caused the trigger state to be set or cleared, in addition to other qualifiers that exist in the Thread or Action's definition.

There are various ways of illustrating the operation of a CorreLog Trigger. One simplified block diagram of CorreLog Trigger operation and message dataflow is shown below.



1. A message is input to the trigger. The trigger compares the message content to a match expression.

2. If the message satisfies the trigger match expression, the trigger enables a gate to permit other messages to be sent to a "Thread" (or an "Action"). This also enables a "count down" timer for the particular trigger.

3. A second message, input to the trigger, can disable the gate, stopping further messages to the thread, and clearing the timer.

4. When the timer expires, this also clears the trigger, disables the gate, and stops further messages from being sent to the Thread, and Action.

Note that the message gate, depicted above, is controlled by the trigger, and is opened when a specific message is received, and closed when a second message is received or the trigger timer expires. While the gate is opened, the "Thread" component can receive and catalog messages.

Trigger Data Items

To support the above construct, each Trigger has specific data items configured on the "Correlation > Triggers" screen. Clicking on the "AddNew" or "Edit" button of this screen permits the user to specify or modify the following data items.

- **Trigger Name.** This is the name for the trigger. The name appears in the "Match Trigger" dropdown menu of the "Threads" and "Actions" screen. The trigger name also is the name of the global variable that is created for the system, as discussed in later paragraphs. Each trigger name must be unique, and under ten characters.

- **Trigger "Set" Expression.** This is a correlation match expression, as discussed in Chapters 2 and 3. The expression can consist of keywords, phrases, wildcards, logical and comparison operators, etc. When an incoming message matches this expression, the trigger is set**.**

- **Trigger "Clear" Expression.** This is an optional field consisting of a correlation match expression, which can be used to clear the trigger prior to the expiration time. When an incoming message matches this expression, and if the trigger is set, then the trigger is cleared and the trigger internal timer is reset.

- **Trigger Expiration Time**. This is a time, in seconds. When the trigger is set, the internal trigger timer starts and continues until the "Clear" expression is received. If this internal timer reaches its expiration time before a clear expression is received, the trigger is cleared. The time can range from 10 seconds to approximately 24 hours.

- **Trigger Expiration Severity.** This is the severity of the internal message that is sent to CorreLog should a trigger reach its expiration time before a "Clear" message is received. By default this value is disabled. The operator can assign some severity to this value in order to watch for events that do not happen, such as an invalid login that is not followed within two minutes by a valid login.

- **Retrigger Flag.** This special flag (either "Yes" or "No") governs how the trigger behaves when it matches a message and the trigger is already set. The default value is "Yes", to retrigger the trigger. If the trigger is already set, and a message matches the trigger set expression, than the internal trigger timer is restarted. Otherwise, the internal trigger timer continues and the trigger ignores this second message.

When CorreLog receives any message, the message content is compared to each trigger specified on the system. This occurs before to checking any Threads or Actions on the system. If the message matches any of the trigger "Set" expressions, that trigger state is set, and the expiration timer starts. If the message matches any of the trigger "Clear" expressions, that trigger state is cleared, and the expiration timer is reset.

Each second, the CorreLog system checks the timers of all currently set triggers. Any trigger that is set, and has expired, is cleared. Therefore, if no explicit "Clear" expression is specified, the trigger will eventually expire and clear itself.

When To Use Triggers

One obvious use for triggers is in providing a visual indication of whether a particular message type has been received. The top-level "Correlation > Triggers" screen updates automatically, approximately every ten seconds, and shows the current state of each trigger, along with the current time to expire. This is surprisingly useful in obtaining the current state of the system and type of network messages received.

In a more general sense, Triggers are required whenever "context" or "state" is needed to fully understand the meaning of received messages. Typical examples of when triggers are useful:

- **Capturing Messages After Startup Events.** One use of triggers has already been mentioned, which is to identify messages associated with system startup. For example, you may wish to capture any errors that occur several minutes after a node reboot, application restart, and identify these in a particular thread.

- **Capturing Messages After Significant Events.** Triggers are useful in capturing the messages related to significant events, such as following an alert. In particular, because an alert can send a "Clear" message, you can start the trigger when the alert is generated (due to some threshold violation) and then clear the trigger when the alert is cleared (when the threshold returns to within limits.)

- **Capturing Files.** Some Syslog systems send full files on error or startup or in other situations. Using CorreLog Triggers, an entire file can be captured in a thread. The start of the file transmission is the trigger, and the trigger expires after 10 seconds or so, which is ample time to capture these types of transmissions. Without triggers, there is no reliable way to capture and catalog these types of transmissions.

- **Capturing Data Dumps.** Similar to capturing files, triggers permit capturing of data dumps in a thread. Unix systems occasionally send data dumps via Syslog. In this case, the trigger is usually started with a keyword such as "dump started", and cleared with a keyword such as "dump ended".

- **Detecting "Non-Occurring" Events.** Triggers provide a method of detecting when expected events do not happen. For example, if you expect some message to occur within some fixed time following some other message, you can set the "Trigger Expiration Severity" to some value, which will generate an internal log message should the trigger reach its expiration time before the second message is received. This internal "now expired" type message can be further correlated like any other message.

Using Global Variables In Triggers

If the user examines the data items associated with a trigger, certain items seem conspicuously absent. In particular, unlike "Threads" and "Actions", there appears to be no way to qualify a trigger based upon the device name, time of day, facility, etc. Also, there appears to be no way to qualify one trigger with another trigger.

This perception is untrue. The message device address, facility, severity, and time of day can all be tested as part of the match and clear expressions, using global variables. For example, the user can qualify a Trigger set or clear pattern with a comparison operator such as "($address eq 10.1.2.1) and ($time lgt 08:00) and ($time lle 18:00)". This type of expression can be in addition to matching specific patterns.

Therefore, Triggers can be fully qualified with expressions to achieve the same level of specific message targeting as Threads and Actions. The only difference is that these qualifiers are embedded into the "Set" and "Clear" match expressions.

One reason for not including these fields is to keep things simple for those users who are looking only for the basic trigger functionality (which by itself is quite powerful) This avoids forcing the user to specify Match Address, Match Time, etc., fields for both the "set" and "clear" patterns, which would greatly increase the number of fields associated with each trigger.

CorreLog triggers typically consist of either very simple constructs, needed to establish a general context, or highly targeted constructs that require more than simple qualifiers available on the Actions and Threads screens.

Trigger Specific Global Variables

Triggers themselves create global variables, which "latch" message data when they are set. Each trigger has a set of global variables associated with it. These are similar to the global variables listed in Chapter 3 of this guide, as follows:

- **"$(triggername)" Global Variable.** A global variable corresponding to a trigger name, when found in any correlation expression, is replaced with the current state of the trigger, either the keyword "set" or the keyword "clear". For example, if a trigger exists with the name "MyLatch", then the comparison ($mylatch eq set) evaluates to true if the trigger is set.

- **"$(triggername).N" Global Variable.** A global variable corresponding to a trigger name, followed by a number (N) is replaced by the Nth word in the message that set the particular trigger. For example, to test to see if the third word of the message that triggered the "MyLatch" trigger contains the keyword "login", the user can specify the comparison (login in $mylatch.3).

- **"$(triggername).address" Global Variable.** A global variable corresponding to a trigger name, appended with ". address", when found in a correlation expression, is immediately replaced by the IP address of the device that set the trigger. For example, to see if the device that set the "MyLatch" trigger has the same IP address as the current message, the user can specify the comparison ($mylatch.address eq $address).

- **"$(triggername).facility" Global Variable.** A global variable corresponding to a trigger name, appended with ".facility", when found in a correlation expression, is immediately replaced by the facility of the message that set the trigger. For example, to see if the facility of the message that triggered "MyLatch" was equal to "internal", the user can specify the comparison ($mylatch.facility eq internal).

- **"$(triggername).severity" Global Variable.** A global variable corresponding to a trigger name, appended with ".severity", when found in a correlation expression, is immediately replaced by the severity of the message that set the trigger. For example, to see if the severity of the message that triggered "MyLatch" was not equal to "debug", the user can specify the comparison ($mylatch.severity ne debug).

- **"$(triggername).facnum" Global Variable**. A global variable corresponding to a trigger name, appended with ".facnum", when found in a correlation expression, is immediately replaced by the facility number of the message that set the trigger. This will be the numeric value of the trigger message facility, ranging from "0=Kernel" to "24=Other". (User defined facilities, which by their nature have no numeric value, are assigned a number of 24, following the last official facility number of 23=Local7) For example, to see if the severity of the message that triggered "MyLatch" was a "localN" facility, the user can specify the comparison ($mylatch.facnum ge 16).

- **"$(triggername).sevnum" Global Variable.** A global variable corresponding to a trigger name, appended with ".sevnum", when found in a correlation expression, is immediately replaced by the severity number of the message that set the trigger. This will be the numeric value of the trigger message severity, ranging from "0=emergency" to "7=debug". (Note that the Syslog protocol defines the highest severity to be zero). For example, to see if the severity of the message that triggered "MyLatch" was greater than info, the user can specify the comparison ($mylatch.sevnum lt 6).

- **"$(triggername).date" Global Variable.** A global variable corresponding to a trigger name, appended with ".date", when found in a correlation expression, is immediately replaced by the date of the message that set the trigger, in "YYYY/MM/DD" format. For example, to see if the date of the message that triggered "MyLatch" had the same date as the date of the current message, the user can specify the comparison ($mylatch.date eq $date).

- **"$(triggername).wday" Global Variable.** A global variable corresponding to a trigger name, appended with ".wday", when found in a correlation expression, is immediately replaced by the weekday abbreviation of the message that set the trigger, either "mon", "tue", "wed", "thu", "fri", "sat", or "sun". For example, to see if the date of the message that triggered "MyLatch" had the same date as the date of the current message, the user can specify the comparison ($mylatch.wday eq $wday).

- **"$(triggername).time" Global Variable.** A global variable corresponding to a trigger name, appended with ".time", when found in a correlation expression, is immediately replaced by the time abbreviation of the message that set the trigger, in 24-hour format ranging from 00:00:00 to 23:59:00. For example, to see if the date of the message that triggered "MyLatch" occurred at noon, the user can specify the comparison (12:00:?? in $mylatch.time).

Note that whenever a user adds a trigger, the above global variables immediately become accessible.

Also, as with all global variables, if the user references a trigger with a global variable that no longer exists, the variable name is not substituted in the expression. This implies that, if a trigger is deleted, those threads and actions depending upon the trigger become inoperable.

Users should be aware: although checks are applied to list the dependencies of triggers (preventing triggers from being deleted if they have dependencies) these checks are not applied to the trigger names used in match expressions. Therefore, deleting a trigger that is used only in an expression may make it impossible to ever match the particular expression.

Using Trigger Global Variables In Actions And Threads

Trigger global variables greatly enhance the power of triggers to qualify specific messages within threads and actions. For example, consider a trigger called "logins", which matches the message associated with a particular login. We wish to capture the username, which always follows the login message in a separate message. The "Thread" match pattern, in addition to collecting messages only after the "logins" trigger is set, can use the match expression ($login.address eq $address), which fires the trigger only when the address of a message matches the address of the message that set the trigger.

Another useful function of global triggers is to permit a "Thread" or "Action" to be qualified by multiple trigger states. For example, if there are four triggers named Trig1, Trig2, Trig3, and Trig4, the user has complete flexibility in qualifying messages based upon these four triggers, such as with the match expression: "(($trig1 eq set) and ($trig2 ne set)) or (($trig3 eq set) and ($trig4 ne set)). Note that these trigger states are incorporated into the match expression, still leaving the "Match Trigger State" field in reserve for other uses.

Cascading Multiple Triggers

Usually, it is not necessary to have one trigger rely on the state of another trigger. However, it may be the case that you want to capture a message only after "Pattern A", occurs, then "Pattern B" occurs, etc.

Although no explicit "Match Trigger State" field occurs on the Trigger configuration screen, you can still make a trigger dependent on one or more triggers through the use global variables in match expressions. The user defines a trigger, for example "PattA". Then, the user defines a second trigger, for example "PattB". As part of the "Set Expression" value of the second trigger, the user specifies ($patta eq set), which insures the "PattB" trigger is set only if the "PattA" trigger is set first.

Using this technique, deep context can be established, where various messages must occur in a specific sequence before a particular thread is updated with a message, or a particular action is executed. Experience shows that providing careful naming conventions to triggers makes this activity much easier to analyze and keep track of. Otherwise, trigger associations can quickly becoming complex and difficult to understand, especially for analysts trying to understand the trigger rules created by some other user.

Section Summary, And Additional Notes

1. Triggers provide the ability to add "context" to correlation, and operate as both "gates" for messages, and "latches" to store message information.

2. Each trigger accepts a "set pattern", and "expiration time", and an optional "clear pattern".

3. When a message matches the "set pattern", the trigger is set and an expiration timer is started.

4. When the timer expires, or if a message matches the optional "clear pattern", the counter is cleared.

5. The "Trigger Expiration Severity" value can be used to log a message should a trigger expire. This can be used to detect the "non-occurrence" of some event, such as waiting for a valid login within one minute of an invalid login. The "trigger now expired" message is logged as any other message (at the user specified severity) and can be correlated like any other message.

6. Triggers can "gate on" the collection of messages by "Threads", and the execution of programs by "Actions".

7. Triggers support a special type of global variables that permit "Threads" and "Actions" to work with multiple triggers, and for triggers to be cascaded to add deep context.

8. Using global variables, triggers can be cascaded, so that one trigger is set only if some other trigger is set. This can establish deep context in messages, where multiple message must occur (perhaps in a specific order) before a trigger is actually set.

Section 6: Correlation Macros

As shown in previous sections, correlation match expressions can range from simple keywords to fairly involved expressions. In practice, these expressions are often reused, with minor modifications, in Threads, Actions, and Triggers.

To simplify this reuse of correlation expressions, CorreLog provides a traditional "Macro" facility, where a keyword can represent an entire expression. This simplifies the creation and implementation of match expressions throughout the system.

The "Correlation > Config > Macros" screen provides general utility in creating, editing, and deleting macro correlation expressions that can be used in various correlation rules on the system. Once a macro is defined, it is referenced by a Thread, Action, or Trigger by bracketing the macro name with the special "double at" @@ characters, such as "@@my_macro@@". During execution of the expression, the macro name is then replaced by its corresponding value.

In addition to simplifying the implementation, organization, and maintenance of correlation match expressions, the Macro facility provides a unique perspective on correlation requirements by permitting a user to think in higher terms than discrete correlation terms, as described in this section.

Correlation Macro Function Defined

The term "Macro", when applied to software systems, refers to a single short keyword, called the "macro name", which is used to represent a longer phrase, called the "macro value". During execution (or just prior to execution) each macro name is replaced by its macro value.

CorreLog employs a straightforward macro facility, where macro names all take the form "@@macro_name@@". The macro names are case-insensitive, can contain alphanumeric characters and underscores, but cannot contain spaces. Each macro name on the system must be unique, and macros cannot be nested.

Macro values, associated with each macro name, do not follow any particular conventions. The values can be any text string or data, and are unconstrained except for a length of 500 characters or less. The value of the macro can be a single word, a phrase, or a full or partial match expression that implements all the rules and conventions presented in previous sections.

Note that the "double at" signs, used to distinguish macro names, make it highly unlikely that any text string will ever match the macro name; hence macros can almost always be used without any special regard or consideration for being confused with message content.

Note that CorreLog also employs macros as part of its Sigma Framework, which permits web pages to be served containing macro values. These macros operate in a fashion almost identical to the macros described here, but have a different purpose, and should not be confused with the correlation macros discussed in this manual.

Macros are configured on the "Correlation > Config > Macros" screen, which permits the user to add, modify, and delete macro definitions. A set of highly useful macros comes with the CorreLog system, and is documented here in a later section.

The "Expression Evaluation" tool, discussed at the end of the first section, automatically expands macros and can be used to test macro values. Likewise, clicking the "Audit" link at the bottom of "Threads", "Triggers" and "Actions" screens shows the match expression with macro values substituted for any macro name.

When To Use Macros

Macros are not required for any correlation to take place. The user can always enter text directly into a "Match Expression" field instead of a macro. However, it will quickly become apparent to any CorreLog user that certain match patterns, required for correlation, are used over and over again within the CorreLog system. In order to promote maintainability of the system, these values should be macros.

By making a particular common pattern into a macro, and using that macro in correlation Threads, Actions, and Triggers, the user can make a global change by modifying the macro value as opposed to modifying the value in multiple locations within the system.

Note that macros are typically used in expressions along with other qualifiers, joined by logical operators to other keywords, phrases, and other macros. For example, a macro called "@@process_terminated@@" may contain keyword needed to identify those messages associated with a process terminating on the system. This macro can then be used in a match expression along with the name of the process, such as @@process_terminated@@ and spoolsv.exe", which matches any time the "spoolsv.exe" process exits. This illustrates an important consideration when defining a macro: the value is quite often ambiguous by itself, but agreeable to further refinement by using the macro with additional expression qualifiers.

It is difficult to over-use macros. Any keyword or phrase or expression that is of any significant complexity is a good candidate for a macro.

Macros, As Semantic "Sensors"

One way of looking at macros is to consider them as able to sense the meaning of a message, at some basic level.

For example, consider a macro called "@@login@@" which corresponds to the expression (login or "logon" or "logged in" or "logged on"). If a message contains any of the keywords represented by the macro, then it is most assuredly a message dealing with a system login of some unspecified type.

The @@login@@ macro described above, by itself, does not provide any additional qualification of the message content. But now consider a second macro called "@@failure@@" which corresponds to the expression (unsuccessful or fail or error or invalid). As before, you can be fairly assured that any message that matches this expression some how indicates a failure of some sort. If a message contains any of these keywords, some type of failure has occurred.

These two macros can be coupled together. If the user implements a match expression of "@@login@@ and @@failure@@", incoming messages will be substantially qualified. Specifically, if the message is sensed as a login, and also a failure, then it is added to the "Login Failures" thread. Any message that matches these keywords will almost certainly represent a login failure of some system.

The correlation expression can be even further qualified, such as with a @@database@@ macro that contains keywords associated with identifying messages from a database. The user can then add this macro (via an "and" logical operator) so that any database login that fails is correctly cataloged and threaded.

Because of the "Cohesive Semantic Similarity" of log messages, these macros do an excellent job of qualifying messages. Although this technique might become cumbersome in programming a general Chatbot, it is quite easy to create correlation expressions that target specific types of messages with the relatively narrow "Message Space" associated with log information.

Once the "@@login@@ and @@failure@@ and @@database@@" correlation expression is satisfied by an incoming message, the message is added to a thread, such as with the title "Database Login Failures". When the number of messages added to this thread meets or exceeds some limit during a specific time interval, this can result in an action, such as a ticket being opened, to indicate that multiple invalid database logins have occurred. For example, this might indicate a possible security threat to the enterprise.

Macros As Synonym Lists

The actual macro value often corresponds to a list of synonyms, conjoined with the "and" logical operator. The macro becomes a synonym for a whole list of words. This can be seen in the "Correlation > Config > Macros" screen, where various default synonym lists exist.

The nature of "semantic correlation" requires easy representation of synonyms. Because the CorreLog system is matching specific meaning, the idea of synonymy is inescapable. Certain different words mean the exact same thing. This is fully supported by macros, where a certain name can represent a variety of different words with identical meanings.

For example, the user might wish to define a macro called "@@crit_procs@@" that serves as a list of the critical process names on the system. Such a macro might have a long list of process names. The value might be something such as "lsass.exe and svchost.exe and smss.exe and csrss.exe and winmgmt.exe". When the user wishes to refer to any of these processes, the user simply specifies the "@@crit_procs@@" macro. Likewise, to update the list of critical process names, the user adds a new process name to the list via the macro editor instead of updating this list at each location the macro is used. In this example, each of the processes is synonymous with the meaning "critical processes".

Once the @@crit_procs@@ macro is defined, the user can create an expression such as "@@crit_procs@@ and @@process_terminated@@" which matches those messages that contain information about a critical process exiting on a system. Likewise, to acquire information about processes that are not critical, the expression is modified to be something such as "@@process_terminated@@ and not @@crit_procs@@"

Standard CorreLog Macros

The following are standard macros for CorreLog. The actual macro values can be inspected or modified via the "Correlation > Config > Macros" screen. Note that this constitutes only the most basic macro set, and that many users will derive more specific macros based upon these more generic macros. All macros that come with the initial installation are prefixed with "gen_" to signify their generic nature.

- **@@gen_admin@@** This macro matches messages associated with system administrators and administrative activities.

- **@@gen_blocked_connect@@** This macro matches messages associated with blocked network connections.

- **@@gen_coldstart@@** This macro matches messages associated with system startup and restarts.

- **@@gen_created@@** This macro matches messages associated with objects (including files and other memory items) being created on the system.

- **@@gen_deleted@@** This macro matches messages associated with objects (including files and other memory items) being deleted or removed from the system.

- **@@gen_denied@@** This macro matches messages associated with failures due to permission problems, or access errors.

- **@@gen_failure@@** This macro is similar to the @@gen_denied@@ macro, except is more targeted for messages that are due to system failures.

- **@@gen_hardware@@** This macro matches messages associated with hardware items on the system, such as printers, keyboards, and peripherals.

- **@@gen_login@@** This macro matches messages associated with login events for Windows, Unix, and application programs.

- **@@gen_logout@@** This macro matches messages associated with logout events for Windows, Unix, and application programs.

- **@@gen_modified@@** This macro matches messages associated with objects being modified on the system.

- **@@gen_protocol@@** This macro matches messages associated with network transmission protocols, such as TCP, IP, UDP, etc.

- **@@gen_storage@@** This macro matches messages associated with computer storage, including the main disk and memory, and removable storage devices.

- **@@gen_success@@** This macro matches messages associated with successful completion of processes or operations.

- **@@gen_virus@@** This macro matches messages associated with viruses and protection programs, security programs and firewall programs.

Macro Name And Value Conventions

Macro names can only be alphanumeric characters and underscores, and are always downcased to lower case letters. Macro values can only be 500 characters or less.(In any match expression, the total number of characters is 2000 characters, which generally permits multiple large macros to be implemented in a single correlation statement.)

Other conventions, although not necessarily strictly required, include the following:

1. The macro name contains a prefix that describes the particular type of macro. All default macros begin with a "gen_" suffix to identify them as generic macros. Other prefixes might include "proc_" for process macros, "unix_" for UNIX macros, "win_" for Windows macros, etc. This helps organize the data.

2. Opening and closing parenthesis usually bracket macro values. While this is not strictly required, it makes the evaluation of the macros and display of these macro values slightly more user friendly. When the macro value is enclosed in parenthesis, the macro value is interpreted as a stand-alone expression.

3. Macro names are typically terse, under 20 characters. This promotes the idea that a macro is best used as a general "sensor". If the macro name is

long, then it typically corresponds to a very specific pattern that may be difficult to reuse. Also, as a side-consideration, long macro names may warp and stretch the CorreLog "Macros" screen in an undesirable way.

Note that macros cannot be nested. If a user needs to match a message that contains a string that is identical to a macro, the user can create a macro for that string, which serves to match some other portion of the target message. This will rarely if every be necessary, because the double at "@@" characters occur infrequently in messages.

For example, to match a text string such as "Macro @@test@@ is undefined" the user will need to define a macro such as "@@macro_ref@@" with a value such as "Macro * undefined", and then use that macro in a system expression.

Unmatched Macros

The CorreLog system does not permit the user to delete a macro without first deleting or modifying each location where the macro is used. (The list of dependents for any macro is available by drilling into the "Edit" screen of a macro and then clicking the "View This Macro's Dependents" hyperlink.) This makes it difficult to have an obsolete reference to a macro that no longer exists.

Should any macro become undefined, such as through external modification of macro fields, the macro name is simply passed as a part of the expression, and is not substituted. This behavior is similar to the way that undefined global variables are handled, although the situation occurs less frequently.

Section Summary, And Additional Notes

1. Macros serve to organize and simplify the creation and maintenance of expressions on the system by allowing easy reuse of correlation match expressions in "Threads", "Actions", and "Triggers".

2. Macros can be thought of as both "sensors" and "synonyms" for message meaning. In particular, creation of synonym lists is an inescapable requirement of "semantic correlation".

3. Although macros are not required to implement any specific correlation, users will quickly see the value of using macros in place of specific expressions. Any expression that is used more than once, or has any significant complexity, is a good candidate for a macro.

4. Macros are often conjoined with other macros or keywords or expressions using the logical operators. Therefore, macros are often used as building blocks for larger correlation strategies.

Section 7: Correlation Applications

This final section puts together various concepts discussed previously by explaining the practical application of the correlation elements introduced in Chapter 1, and elaborated upon in Chapters 2 through 5.

CorreLog has application within diverse business models. The program can operate as a security system, supporting regulatory compliance, but can also serve as an essential building block in a larger management strategy. For example, it is valid to use CorreLog in such diverse roles as a simple Syslog message aggregator, or as an SNMP trap collector, or as a message archiving system, or as an automation component in an incident management system. This section confines the discussion to CorreLog's role as an enterprise "correlation server". However, it is important to consider that this is only one of a multiplicity of possible application roles for CorreLog.

This section also provides a further discussion of the various correlation components, specifically how to effectively use Threads, Alerts, Actions, and Tickets to create a correlation strategy for an enterprise. As part of this discussion, this section suggests practical configuration policies for the system, as well as more theoretical discussions

This section augments the discussion of these correlation components, found in the "CorreLog User Reference Manual" and provides additional notes and concepts not documented elsewhere, which may be useful or required to establish high degrees of customized correlation.

Correlation Server Input and Output

In starting a discussion about correlation applications, it is useful to review again the nonspecific function of CorreLog targeted by this guide, that is: to act as a "correlation server" for an enterprise.

CorreLog is intended to "serve up" correlation, similar to the way that a print server will "serve up" print jobs, or a web server will "serve up" web pages. Although CorreLog has many other useful functions and applications, such as archiving of data, the correlation of data is the main purpose and intent of the program, at least as discussed in this manual

Unlike printouts or web pages, "correlation" is a somewhat more abstract quantity. It is useful to discuss inputs and outputs to the server in order to explain what "serving up" correlation may actually constitute

- **Correlation Inputs.** If we disregard the configuration files of the program as an input, we can say that the only inputs to CorreLog are time stamped messages. These messages represent arbitrary communications, such as error or status messages, informational messages, and also state messages and signals. The input is in the form of Syslog data, where each message has a "facility" and "severity" associated with it, and there exist various adapters (such as the Windows Tool Kit and SNMP Trap Interface) that convert messages of other formats to Syslog messages.

- **Correlation Function**. CorreLog stores and archives these messages, but its main function is to look for certain relationships in these messages and take action when those relationships appear. The actions range from running arbitrary programs, cataloging of data, and opening of tickets.

- **Correlation Outputs.** CorreLog produces reports and message catalogs as part of its tangible output. A more important and interesting output (at least within the context of this discussion) is an "execute" command, which runs arbitrary and varied programs to take action when items are correlated. The CorreLog system can open tickets, send Syslog messages and SNMP traps, send e-mail, and take corrective action when system faults occur. This re-enforces CorreLog's "server" role as a stand-alone component that provides a highly interoperable "service" to an enterprise.

The output of CorreLog, generally speaking, is the execution of some action. In addition to the built in actions of the program (such as opening Tickets) user actions are configurable in the "Correlation > Actions" screen. CorreLog comes with a comprehensive list of action programs as part of its initial baseline, including programs to send e-mail, send SNMP traps, update database items, or send other types of notifications. These actions are documented in detail within the "CorreLog User Reference Manual", along with information on the arguments that accompany program execution.

The various sections that follow concentration on the actual correlation function of the system, in terms of the programs inputs and outputs. Further information on these functions is available in the accompanying "Correlog User Reference Manual", including information needed to extend the system with third party scripting languages.

Normalization of Input Data

The first step in CorreLog's processing is normalization of input data. This guide has not explicitly discussed the "Messages" navigation tab of CorreLog, or its functions. However, it is worth noting that this portion of the CorreLog system comprises about half of the entire system functionality.

One of the nice things about CorreLog is that this data normalization process can be trivial, or completely optional. The "semantic correlation" algorithms do not require normalized data. Input data normalization, if implemented, basically consists of configuring optional filters and overrides, which can be used to slightly pre-process the incoming data and perhaps make correlation setup slightly easier.

When messages are received, they are first filtered, and then passed through "address", "facility", and "severity" overrides.

- **Filters.** The user can configure filters in the "Messages > Config > Filters" screen of the system. Each filter consists of a simple keyword or wildcard combination, without spaces. If an incoming message matches any filter, it is removed from the system (and placed in the special "Filter" screen for further reference.)

- **Address Overrides.** After passing all the filters, the message is next processed by the address override component. This component is similar to the filters, but rather than filtering the message, instead the address of the message is modified depending upon a simple keyword or wildcard combination. This may be useful in handling environments that implement Network Address Translation (NAT).

- **Facility Overrides.** After passing through the address override component, the message is next processed by the facility override component. The facility code of the system can be modified depending upon a simple keyword or wildcard combination. This may be useful when correlating messages by facility. (In particular, the user can define new and non-standard facilities as discussed in the next section.)

- **Severity Overrides.** After passing through the facility override component, the message is next processed by the severity override component. The severity code of the system can be modified depending upon a simple keyword or wildcard combination. This may be useful when correlating messages by severity.

Keywords used when filtering and overriding of data does not permit any of the correlation expressions defined in this manual. (This is clearly identified on the filter and override screens.) The only expression permitted is a simple keyword match, with possible wildcards. This is necessary in order to keep the extremely high throughput of the system (which approaches or exceeds 1000 messages per second).

Creating Custom Facilities

A "facility" code is incorporated into each message received by CorreLog. This code can then be used as a qualifier in various locations of the system, especially within "Threads", "Triggers", and "Actions" (but also in other locations) to match the particular source or general classification of the message.

In practice, depending upon the particular business environment, facility codes may not be as relevant as one might thing. In particular, the software programmer who first defines a message is solely responsible for assigning the facility codes, and many of these facility codes have become archaic and deprecated. For example, the "uucp" facility code is fairly obsolete. (In CorreLog, the "uucp" facility has been renamed "network", to more closely match the functionality of these messages.)

Because facility codes may not always be pertinent, CorreLog provides the unique ability to permit users to create their own facility codes, thereby allowing messages to have a facility code such as "Cisco" or "MyAppl", or "MS-Exchange". The user can then set a facility override to assign this facility whenever an incoming message matches a particular criteria.

For example, all the messages from a particular device address that contain a particular keyword may be automatically assigned a facility code "HTTP" to denote that these are HTTP server type messages. The user can then match the message facility rather easily in the "Threads" or "Actions" screen by selecting the "Match Facility" to be HTTP.

This adds an important dimension to the correlation process, which users may find extremely handy. Creating and assigning facility codes to messages achieves an extra layer of correlation, implemented by the CorreLog "Messages" component prior to the more rigorous correlation implemented by the "Correlation" component.

Input Message Filter Policies

With regards to specific policy, it is generally a good idea to keep input filters simple, and filter data whenever possible at the source. If the user is employing the CorreLog Windows Tool Kit software, filters can be applied remotely at the Windows platform by editing the remote configuration at the main CorreLog Web Interface. (This is accomplished by enabling the remote configuration editor after clicking the device IP address for the Windows platform, wherever this link appears within CorreLog.) If the user is monitoring a Unix platform, filtering can be applied in the Syslog configuration file (usually "/etc/syslog.conf").

If there are only a few messages that need to be filtered, the "Filters" component can make fine adjustments at the main CorreLog server. Generally, filters can cause confusion when used indiscriminately, since certain expected messages can easily be accidentally filtered. To minimize the impact of this happening, CorreLog contains the "Messages > Filtered" screen, which holds the list of filtered messages for the current day, dropping this list each night at midnight.

Thread And Trigger Component

Once messages have been received and logged, the first "official" step of the correlation process is to "Thread" the messages.

The "Threads" and "Triggers" components, and their role in cataloging messages, has already been discussed exhaustively in this manual with regard to match patterns. However, it is worth discussing these components from a slightly different perspective.

The term "Thread" comes from the idea that most human conversation naturally follows a certain sequential pattern. For example, it is common to discuss "e-mail threads" or "discussion threads". Likewise, in social networking websites, it is common to find the term "Thread" applied to bulletin boards and forums. Each thread in a discussion forum has messages that are closely related to a single topic.

The use of the word "Thread" to discuss related messages is taken directly from this idea. In a fashion similar to social networking forums, a CorreLog Thread is a particular topic. The topic can be broad (such as "All Logins") or very narrow (such as "Invalid Logins For HTTP Server"). The particular topic may have context associated with it (supplied to Triggers) such as "Error Messages Following Reboot". As with many social networking sites, the system can contain many different threads, only a few which are "popular", and many of which are seldom if ever used. Finally, as with an online forum, the most recently updated thread is by default automatically pulled to the top of the list making it clear what the active topics are.

CorreLog permits thousands of threads to exist. (The exact number may be licensing dependent, but is typically at least 5000 different threads.) The most recently updated thread is at the top of the list, followed in chronological order by the next most recently updated thread, etc. From the "Correlation > Threads" screen, the user can clearly see system activity, and drill down into the system to view the detailed messages comprising the thread.

Thread Configuration Policies

Arguably, the most important part of a thread is the "Thread Title", which appears on the main "Threads" screen, which identifies the topic of the thread, and which is referenced in the Alerts and Reports components of the system. The thread title can be up to 72 characters, which permits a high measure of description. Thread titles can be modified after they are created, to make them more descriptive and accurate, as required.

There is no check that makes sure the title actually agrees with the match pattern. (This may be a limitation removed in future versions of the program, but will require much more reliable linguistic techniques than exist given the current state-of-art.) It is therefore important to choose thread titles that are descriptive enough to identify the message thread, but general enough to permit minor variations in messages. One way to accomplish this is to incorporate one or more of the match pattern keywords or macro names into the thread title.

With regard to creating threads, the system supports massive numbers of threads. The system also allows one thread to be quickly derived from another thread (using the "SaveNew" button on the "Modify" screen, discussed later in this chapter.) The user should not be afraid to create a large quantity of threads. If a thread is seldom or never used, it drops out of normal view past the bottom of the current page. Therefore, creating a new thread, which is seldom or never used, is actually fairly harmless.

The user can audit the full list of threads via the "Audit" hyperlink at the bottom of the display. This provides a simple way to view all the thread data, and audit whether the title names are actually descriptive of the thread content based upon the configured match qualifiers and match expressions.

Deriving New Threads, Triggers, and Alerts

Although it is a common activity to configure new Thread, Trigger, and Alert components, It is often not necessary to derive a completely new component on the system. The basic installed CorreLog system provides a number of predefined components that work in the general case. If a new type of correlation is required, it is may be sufficient to simply modify these components slightly, adding new qualifiers to these components to make them more specific, and then saving these components under a new name. This is accomplished via the

"SaveNew" button located on the "Threads", "Triggers", and "Alerts" edit screens, which retains the original component and creates a new, modified component.

For example, a user may be interested in any login errors that occur for a specific device or application or both. In that case, rather than creating a new Thread from scratch, the user can edit the "Invalid Logins" thread and simply add a new qualifier to the "Match Expression" and / or "Match Address" fields, needed to distinguish that Thread and make it more specific.

In this case, the user modifies both the Match Expression (or other qualifiers). The user then modifies the Thread Title field to be something such as "Invalid Application Logins." Clicking "SaveNew" button rather than the "Save" button creates a new CorreLog Thread that targets login errors from the particular and explicit application rather than all login errors that have occurred in the enterprise.

This technique works with all correlation components, and allows an administrator to rapidly create large numbers of specialized components starting with existing components. The result is more refined and granular correlation functions.

Viewed in this way, each correlation component serves as a template for deriving other and more unambiguous components, increasing the specificity of the correlation process.

Action Program Configuration

At the beginning of this chapter, "Actions" were identified as the principle output of the system. The "Correlation > Actions" screen permits the user to create various actions, including the execution of predefined actions and customized actions. Execution of actions will be one of the central uses of the CorreLog server. The precise type of action taken by CorreLog depends upon the integration strategy and objectives of the organization.

Action programs simply monitor the input stream of messages, in a fashion identical to threads. When a message occurs, possibly qualified as a complex match expression along with other factors, the action program is immediately executed. The specific arguments to the action program are documented in the "CorreLog User Reference Manual", and are passed to the program as either environmental variables to a batch file, or as command line arguments. These arguments include the message content, device address, facility, severity, and other useful parameters.

The message that triggers the action can come directly from a device, or can come from a configured CorreLog Alert, or any combination of these including triggers.

The "Actions" facility constitutes one of the most extensible features of the system, and opens the door to a number of creative applications. For example, multiple action programs can update multiple relational databases with information needed to support dashboards. (The CorreLog system provides an extensive capability to convert database queries into screens, as documented in the CorreLog Sigma Framework Manual.) Additionally, Action programs can launch recovery programs to take automatic corrective action.

Each time an action program is executed, it maintains a log of its execution. The output of this log is available by clicking the "View Debug Log File", which shows a list of the most recent execution errors. This provides a simple way of testing the validity of any action program.

Advanced Correlation Using Actions

One specific use of "Action" programs is to further correlate data. The "Threads", "Triggers' and "Alerts" components can be used to create correlation rules to target a wide range of messages and applications. However, it is also possible to program a simple "Action" script that satisfies very obscure correlation objectives. For example, a user can correlate messages using the native Regex functions found in Perl programming language, and can look at external data found in files and databases and other data stores needed to perform a unique correlation.

This is a straightforward activity to accomplish. The user simply writes a Perl script (or other type of script), which accepts the message data from environmental variables established by the CorreLog system. The user's script is executed when a message is received, possibly pre-qualified by a match expression or device or other parameter. The script performs the correlation activity, and then sends a Syslog message back into the message stream. The actual Syslog message is generated either by running the "system\sendlog.exe" program from the script, or by using a simple socket interface, such as that documented in the Resources section of the CorreLog website.

The resulting Syslog message is highly significant. It has been processed to accommodate a highly targeted objective. This Syslog message is subsequently handled by the other elements of the system like any other message, such as to run further actions, or to send additional alerts, or to open tickets on the system.

Action Program Configuration Policies

Unlike Threads and Alerts, Action programs should be configured very deliberately. This is due to several potential pitfalls. Each Action program is executed sequentially, hence if a lot of action programs are being executed this can drag down the system performance. Additionally, it is a good idea to be sparing of execution cycles related to automatic (and perhaps unpredictable) execution of programs. A typical example is related to e-mail notifications, where a configuration mistake of the program can easily generate hundreds (or thousands) of unwanted e-mail message.

Action programs can be driven directly by incoming messages. However, it is a generally a good policy to drive most of the action programs from Alert generated messages. This ensures that programs are only executed occasionally, such as when an alert is set, then expires, then is set again. Although this is not strictly required, it is left to the experience and judgment of the end-user as to whether this is needed for a particular action program.

Specifically, when sending e-mail, controlling the action program by an alert with a test interval of 300 seconds will prevent any further e-mail messages being sent until the 300-second test interval has elapsed. This will reduce the number of e-mail messages to some desirable amount.

Finally, if an action program requires a lot of execution time, such as to perform copious number crunching or database queries, it is probably better to schedule the majority of the program's execution as a Microsoft "Scheduled Task" (configurable in the Control Panel of Windows machines.) The scheduled program can run periodically, at its leisure, and the action program can check the results of this scheduled program periodically to finish final processing of the data.

For example, consider a situation where the user wants to check a huge database for information, and then send-email based upon a received message and the results of that query. Rather than have the action program run a time-consuming database query each time the message is received, it is much more efficient to check the results of the query (executed every 10 minutes as a scheduled task by a separate program) and then take action based upon the pre-processed query results and the received message. This permits the action program to be executed dozens of times each second, or more, rather than at the much slower rate that is dependent on the possibly long query time of the database.

Ticket Component

As mentioned earlier, one of the major outputs of the system are "Tickets", which indicate that a threshold has been violated. This is a special built-in action, which can be viewed as the highest level of correlation on the system, since it correlates one more item into the mixture, that being an "Assignee" for the incident.

Tickets contain the text associated with an alert message, and also reference the original messages that caused the alert to be sent. Each ticket, in addition to having a facility and severity, also has an assignee. There are two types of ticket assignees:

- **CorreLog User Assignee.** Tickets can be assigned to a registered CorreLog user who is actually responsible for resolving the ticket. For example, some users might be solely interested in firewalls, while other users work strictly with Windows platforms. This assignee will be a human being or group that process and resolves the ticket.

- **Ticket Group Assignee.** Tickets can be assigned to arbitrary functional groups created by the administrator. These ticket groups can be associated with device types, applications, or specific network devices. These ticket groups can also be human beings, but more typically they may be arbitrary identifiers that serve as the final cataloging and categorization of correlated data.

In particular, "Ticket Groups" can provide an even higher level of correlation than Threads, since they work both with the match patterns of Threads and Triggers and also the thresholds of Alerts. By working in concert with all of these components, highly meaningful and actionable data can be created.

To implement a ticket group manually, the user creates the Ticket Group via the "Tickets > Config > Ticket Groups" screen. The user then optionally creates a Thread and an Alert for that thread, which is responsible for actually opening the ticket. To assist in creating Ticket Groups, the CorreLog system provides a simple wizard program that allows a user to create all of these components together. The wizard is launched by clicking the "Wizard" button of the "Tickets > Config > Ticket Groups" screen, which guides the user through the process of creating a complete correlation rule consisting of Thread, Alert, and ticket.

Note that this wizard, while useful, is somewhat limited. It is best used to create relatively simple correlation of events that can be identified through a single match pattern and a single alert.

Ticket Handling Policies

More than any other component, the handling of Tickets is highly policy driven by the enterprise rules and objectives. For example, a typical installation may require that every opened ticket be dispatched by a user within a certain time frame. Or, the policy may be such that tickets never have to be dispatched or even inspected or closed. The CorreLog system requires no specific policy, and supports either policy. Once the ticket has been opened, it is up to the enterprise to determine what to do with that ticket, if anything.

In many ways, the effectiveness of the correlation system can be completely gauged by the number and pertinence of the tickets that are being opened. It is not uncommon to have a great many tickets opened on the system. Nor is it uncommon to have only a few or no tickets opened. However, if the opened tickets are not pertinent, then the system has been configured too loosely.

If the user finds a large number of tickets being opened without much value, then the threshold of the alert that controls the ticket needs to be adjusted. This is a common situation, especially when configuring a new alert. The CorreLog system specifically supports this activity, as follows:

1. On the "Tickets" screen, the user can drill down on the "Source Alert Def" hyperlink of the ticket to access the alert that opened the ticket. This screen reflects the data of the source alert definition, and includes an "Edit" button to permit certain parts of the alert to be modified.

2. The user clicks the "Edit" button to edit the source alert definition. From the resulting screen the user can then adjust the alert threshold one or two counts higher, or expand the sample interval. This will make the alert less sporadic and open viewer tickets.

3. The user may also wish to modify the severity of the ticket to decrease its pertinence, reassign the ticket to some other group or user, or completely disable the alert from generating a ticket.

Using this technique, the CorreLog system will rapidly stabilize into a state where only pertinent tickets are being opened, all tickets are meaningful, and the ticket display becomes the trusted source for reporting correlation incidents of the enterprise.

Tickets As Final Outputs

To conclude this section, it should be mentioned that CorreLog tickets, because they represent the highest level of correlation, can be considered the final outputs of the CorreLog system. At some sites, the automated opening of tickets based upon security (or other) incidents will be the only purpose of CorreLog, and the sole service output by the system. For example, ordinary users may relate to CorreLog only through their third-party incident management system, and may be permitted to view only those tickets that have been assigned to them.

CorreLog supports this special role a variety of ways. An administrator can interface the Correlog Server to an incident management system via the system "TICKET.bat" file, can export just the Tickets screen to ordinary users, and can export ticket statistics to a Common Management Database (CMDB). These applications are described in more detail within the "User Reference Manual", and may require CorreLog Profession Services to implement for most non-trivial applications.

Appendix: Reference Tables

This section provides reference tables, and is intended as a quick reference to the various discussions in this manual as follows:

- **Rules For Basic Correlation Expressions.** This table provides the basic rules that apply to correlation match expressions, documented in Chapters 2 and 3 of this guide

- **List Of Logical Operators.** This table provides the list and formal definition of the CorreLog logical operators that can used to join sub-expressions into larger expressions. This table is further documented in Chapter 2 of this guide.

- **List of Compare Function Operators.** This table provides the list and formal definition of the CorreLog comparison function operators that are used within expressions to test global variables. This table is further documented in Chapter 3 of this guide.

- **List of Global Variables.** This table provides the list and formal definition of all global variables, used with compare functions to perform detailed tests against the current message or triggers. These variables and their usage are further documented in Chapters 3 and 4 of this guide.

- **List of Alert Formula Counter Names.** This table provides a description of the counter names associated with each system counter that can be used within "Alert Formulas", described in Chapter 6 of this guide.

Rules For Basic Correlation Expressions

Case Insensitivity	Correlation expression matches are always case-insensitive, without exception.
Simple Keyword Matches	Correlation expressions can consist of simple keywords, which match any portion of the message. For example "su" matches "success" or "super user".
Phrase Matches	If the correlation expression contains spaces, it must be quoted. For example "super user" matches "super user" and "super users".
Full Word Matches	To match a full word instead of a partial match, precede and follow the match expression by a single space. For example " su " matches "the su user logged in", "su login, and "login by su". In the expression, the leading space matches a space or the beginning of line; the trailing space matches a space or the end of line.
Wildcards	A keyword can contain a "*" asterisk wildcard to match zero or more characters, and a "?" wildcard to match a single character. For example "test*fail" matches "the test failed" and "the test did not fail" and also "testfail". Likewise, "test?pass" matches "test passed", or "test-pass", but not "testpass".
Logical Operators	To join expressions into a larger expression, use the "and", "or", "xor" and "not" logical operators. For example "test and not fail" matches any message that contains the keyword "test", and not the keyword "fail", anywhere in the line
Default Logical And	If an expression is composed of several sub-expressions without a logical operator, they are joined by an implied "and", and each expression must match the message. For example, while "super user" matches the specified phrase, if the double quotes are omitted, then this is equivalent to "super and user", and the message must contain both keywords in any order.
Parenthetical Nesting	The user can specify precedence of evaluation using parentheses, which can be deeply nested. For example "(test and file) or (system and user)" matches any message containing both "test" and "file", or any message containing "system" and "user".

Logical Operators

Logical operators join two sub-expressions together. These sub-expressions can consist of keywords, wildcards, phrases, or other expressions, possibly parenthetically nested to change the order of evaluation.

"and" operator	This is the default logical conjunctive operator. For example, "XX and "YY" will match any message that contains both "XX" and "YY", in any order within the message. Both the left and the right operands to the "and" operator must be present somewhere within the message.
"or" operator	This is the logical "or" operator. For example, the correlation match expression "AA or BB or CC" matches the message if it contains any of the three keywords, in any order within the message. Either the left or the right operands to the "or" operator, or both operands, must be present somewhere within the message.
"xor" operator	This is the logical "exclusive or" operator, which matches the message if either the left or right operands appear in the message, but not both operands. For example, "QQ xor RR" matches the message "value of qq", and matches the message "value of rr", but does not match the message "value of qq rr" or the message "rrqq exists". The "xor" operator is not used that often, however is invaluable when actually required.
"not" operator	This is the logical negation operator, which indicates that the keyword or phrase following the operator must not match. For example, "not ZZ" matches any message that does not contain the keyword ZZ. Likewise, the correlation match expression "not AA and not BB and not CC" matches any message that does not contain the all three of the specified keywords, and the correlation match expression "not AA or not BB or not CC" matches any message that contains any of the specified keywords.

The "and", "or", and "xor" operators each require left and right arguments. The "not" operator requires only a right argument. The associative, distributive, and redundancy laws of Boolean logic are strictly followed, as expected. For example, "(not AA) and (not BB)" is the same as "not (AA or BB)".

Comparison Functions

Comparison functions can be used in conjunction with global variables to test fields, device names, and other specialized functions. (See next section for description of global variables.) Comparisons are typically joined with other expressions with logical operators, described earlier. Each operator below requires a left and right value.

"eq" operator	The left value must precisely equal the right value, ignoring any letter case. For example, the expression ($4 eq test) returns true if the fourth word of the message is "test" or "TEST", but not "TestCase". Likewise, the expression ($2 eq $5) returns true if the second and fifth words are the same in the message. The comparison is case insensitive and wildcards are treated as ordinary characters.
"ne" operator	The left value must be different from the right operator, ignoring any letter case. For example, the expression ($4 ne test) returns false if the fourth word of the message is not "test" or Test", but returns true if the fourth word is "TestCase". Likewise, the expression ($2 ne $5) returns false if the second and fifth words are the same in the message. The comparison is case insensitive and wildcards are treated as ordinary characters.
"lt" operator	The left and right values must both be numbers, or begin with numbers. Returns true if the left value is less than the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression ($4 lt 199) returns true if the fourth word of the message is numerically less than 199, or if the fourth word of the message is not a number.
"le" operator	Similar to the "lt" operator, except the comparison is less than or equal to. The left and right values must both be numbers, or begin with numbers. Returns true if the left value is less than or equal to the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression ($4 le 199) returns true if the fourth word of the message is 199 or less, or if the fourth word of the message is not a number.

Comparison Functions (continued)

"gt" operator	The left and right values must both be numbers, or begin with numbers. Returns true if the left value is greater than the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression "($4 gt 199)" returns true if the fourth word of the message is numerically greater than 199, and is also a number.
"ge" operator	Similar to the "gt" operator, except the comparison is greater than or equal to. The left and right values must both be numbers, or begin with numbers. Returns true if the left value is greater than or equal to the right value. If either the left or right values are not numbers, they are regarded as a numeric zero. For example, the expression "($4 ge 199)" returns true if the fourth word of the message is 199 or greater, and is also a number.
"llt" operator	This performs an alphabetical comparison of the left and right values. Returns true if the left value is alphabetically less than the right value, ignoring case. For example, the expression "($1 lt T)" returns true if the fourth word of the message is "success" or "Success", and returns false if the fourth word is "Test", or "workgroup"
"lle" operator	Similar to the "llt" operator, except the comparison is alphabetically less than or equal to. Returns true if the left value is alphabetically less than or equal the right value, ignoring case. For example, the expression "($1 lle T)" returns true if the fourth word of the message is "success" or "TEST", and returns false if the fourth word is "workgroup" or "Workgroup".
"lgt" operator	This performs an alphabetical comparison of the left and right values. Returns true if the left value is alphabetically greater than the right value, ignoring case. For example, the expression "($1 lgt T)" returns true if the fourth word of the message is "workgroup" or "Workgroup", and returns false if the fourth word is "testcase" or "success".

Comparison Functions (continued)

"lge" operator	Similar to the "lgt" operator, except the comparison is alphabetically greater than or equal to. Returns true if the left value is alphabetically greater than or equal the right value, ignoring case. For example, the expression "($1 lge T)" returns true if the fourth word of the message is "test" or "TEST" or "Workgroup", and returns false if the fourth word is "success" or "Success".
"in" operator	This operator tests to see if the keyword or wildcard contained in the left value is found in the right value. This is similar to the correlation expressions discussed previously, but is confined to the value contained in the right value. For example, the expression "(key" in $5)" returns true if the fifth word of the message is "key", "keyword" or "PASSKEY". Likewise, the left value can contain wildcards. For example the expression "(key*o*d in $3)" matches the third word if it is "Keyword" or "keyboard".
"not in" operator	The same as the "in" operator, except that the comparison returns true if the left value keyword or wildcard is not in the right value. For example, the expression "(key" not in $5)" returns false if the fifth word of the message is "key", "keyword" or "PASSKEY", and the expression "(key*o*d not in $3)" returns false if the third word if it is "Keyword" or "keyboard". Note that this is the only comparison function containing two separate words.

Global Variables

These values, which are all preceded with a '$' character, are set by the system each time a message is received (or a trigger is set, in the case of trigger global variables) Global are frequently used with the comparison operators (listed earlier) to perform advanced correlation of messages.

$N	A number, preceded with a "$" character, when found in a correlation expression, is immediately replaced by the word in the current message of the corresponding position. For example, the expression ($3 eq "test") evaluates as true if the third word in the message is the specific word "test". This allows a user to test for a value of a particular keyword based upon the position of the word expected in the message. If the value of "N" is higher than the number of words, no substitution of the $N value takes place.
$address	The word "$address", when found in a correlation expression, is immediately replaced by the IP address of the device that sent the message. For example, if the user includes "$address" as a keyword in the expression, the IP address of the device that sent the message must appear somewhere in the message. More commonly, this value is used in a comparison function such as ("10.1.2.2" eq $address), which matches only if the device that sent the message has an address of "10.1.2.2". This provides an alternative to specifying an address in the "Match Address" specification of thread and action screens.
$username	The word "$username", when found in a correlation expression, is immediately replaced by the username contained in the message (if any) or the string "none" if the message did not contain a username. This value can be used in a comparison function such as ($username eq "jsmith"), which matches only if the keyword "jsmith" is in the message, and "jsmith" is also a username defined in the "Messages > Users" screen.

Global Variables (continued)

$userdata	The word "$userdata", when found in a correlation expression, is immediately replaced by the user group information residing in the "./config/userdata.cnf" file corresponding to the specified username. The username for the message (if any) is used to index the user data. This permits correlation on user information that is external to the message, such as the user's full name, location, e-mail address, or any other data in this configuration file. For example, the value can be used in a comparison function such as ("New York" in $userdata), which matches if the username, specified in a message, has "New York" as part of the $userdata record.
$devname	The word "$devname", when found in a correlation expression, is immediately replaced by the device name associated with the IP address (if any). The device name is the value configured in the "Device Information" screen by the operator, which may or may not be the official DNS name for the IP address. For example, the value $devname eq "localhost" matches only if the IP address of the device that sent the message has a name "localhost" defined in the "Device Information" screen.
$devdata	The word "$devdata", when found in a correlation expression, is immediately replaced by the device group information residing in the "./config/devdata.cnf" file, corresponding to the specified address. The address of the message is used to index the device data. This permits correlation on device information that is external to the message, such as the purpose of the device, its location, its asset information, or other information. For example, the value can be used in a comparison function such as ("Solaris" in $devdata), which matches if the IP address, specified in a message, has "Solaris" as part of the $devdata record

Global Variables (continued)

$facility	The word "$facility", when found in a correlation expression, is immediately replaced by the facility of the current message. This will be either the official facility name, such as "kernel", "user", "network", or can be a user-defined facility. (User defined facilities are discussed in the "CorreLog User Reference Manual", and are configured under the "Messages" navigation tab of the system.) This global variable provides an alternative to specifying a facility in the "Match Facility" specification of thread and action screens, and additionally permits a range of facilities to be specified as part of a match pattern.
$severity	The word "$severity", when found in a correlation expression, is replaced by the severity name of the current message. This will be the official severity name, ranging from "debug" to "emergency". This global variable provides an alternative to specifying a severity in the "Match Severity" specification of thread and action screens. This value, while useful, is generally not as powerful as the "$sevnum" global variable, which is the corresponding numeric value of the severity, and which permits the "lt", "le", "gt", and "ge" numeric comparisons to be made on a severity value.
$facnum	The word "$facnum", when found in a correlation expression, is replaced by the facility number of the current message. This will be the numeric value of the message facility, ranging from "0=Kernel" to "24=Other". (User defined facilities, which by their nature have no numeric value, are assigned a number of 24, following the last official facility number of 23=Local7). This value permits the "lt", "le", "gt", and "ge" numeric comparisons to be made on a facility value.

Global Variables (continued)

$sevnum	The word "$sevnum", when found in a correlation expression, is replaced by the severity number of the current message. This will be the numeric value of the message severity, ranging from "0=emergency" to "7=debug". (Note that the Syslog protocol defines the highest severity to be zero). This value permits the "lt", "le", "gt", and "ge" numeric comparisons to be made on the received message severity value. For example, this permits an expression such as "($sevnum ge 3) and ($sevnum le 6) " which matches any message with severities of "info", "notice", "warning" and "error".
$date	The word "$date", when found in a correlation expression, is replaced by the current date, in "YYYY/MM/DD" format. This may be useful when configuring correlation patterns specific for particular days and / or months, especially when used with the "in" and "not in" comparison operators. For example, the expression ("/??/01" not in $date) will match a message only if it is not the first day of the month.
$wday	The word "$wday", when found in a correlation expression, is replaced by the current three letter weekday abbreviation, either "mon", "tue", "wed", "thu", "fri", "sat", or "sun". This may be useful when configuring correlation patterns specific for particular days of the week. For example, the expression "($wday ne tue)" will match a message only if it is not received on a Tuesday.
$time	The word "$time", when found in a correlation expression, is replaced by the current time, in "HH:MM:SS" 24 hour format. This may be useful when configuring correlation patterns that match specific times, extending the "Match Time" specification of thread and action screens. This global variable is typically used with a lexical compare function, such as "llt" or "lgt". For example, the expression "$time lgt 23:00:00) and ($time llt 23:30:00)" matches a message only if it is received after 11:00 PM before 11:30 PM.

Global Variables (continued)

$(triggername)	A global variable corresponding to a trigger name, when found in any correlation expression, is replaced with the current state of the trigger, either the keyword "set" or the keyword "clear". For example, if a trigger exists with the name "MyLatch", then the comparison ($mylatch eq set) evaluates to true if the trigger is set.
$(triggername).N	A global variable corresponding to a trigger name, followed by a number (N) is replaced by the Nth word in the message that set the particular trigger. For example, to test to see if the third word of the message that triggered the "MyLatch" trigger contains the keyword "login", the user can specify the comparison (login in $mylatch.3)
$(triggername).address	A global variable corresponding to a trigger name, appended with ". address", when found in a correlation expression, is immediately replaced by the IP address of the device that set the trigger. For example, to see if the device that set the "MyLatch" trigger has the same IP address as the current message, the user can specify the comparison ($mylatch.address eq $address).
$(triggername).facility	A global variable corresponding to a trigger name, appended with ".facility", when found in a correlation expression, is immediately replaced by the facility of the message that set the trigger. For example, to see if the facility of the message that triggered "MyLatch" was equal to "internal", the user can specify the comparison ($mylatch.facility eq internal).
$(triggername).severity	A global variable corresponding to a trigger name, appended with ".severity", when found in a correlation expression, is immediately replaced by the severity of the message that set the trigger. For example, to see if the severity of the message that triggered "MyLatch" was not equal to "debug", the user can specify the comparison ($mylatch.severity ne debug).

Global Variables (continued)

$(triggername).facnum	A global variable corresponding to a trigger name, appended with ".facnum", when found in a correlation expression, is immediately replaced by the facility number of the message that set the trigger. This will be the numeric value of the trigger message facility, ranging from "0=Kernel" to "24=Other". For example, to see if the severity of the message that triggered "MyLatch" was a "localN" facility, the user can specify the comparison ($mylatch.facnum ge 16).
$(triggername).sevnum	A global variable corresponding to a trigger name, appended with ".sevnum", when found in a correlation expression, is immediately replaced by the severity number of the message that set the trigger. (Note that the Syslog protocol defines the highest severity to be zero). For example, to see if the severity of the message that triggered "MyLatch" was greater than info, the user can specify the comparison ($mylatch.sevnum lt 6).
$(triggername).date	A global variable corresponding to a trigger name, appended with ".date", when found in a correlation expression, is immediately replaced by the date of the message that set the trigger, in "YYYY/MM/DD" format. For example, to see if the date of the message that triggered "MyLatch" had the same date as the date of the current message, the user can specify the comparison ($mylatch.date eq $date).
$(triggername).wday	A global variable corresponding to a trigger name, appended with ".wday", when found in a correlation expression, is immediately replaced by the weekday abbreviation of the message that set the trigger, either "mon", "tue", "wed", "thu", "fri", "sat", or "sun", for example, the user can specify the comparison ($mylatch.wday eq $wday).
$(triggername).time	A global variable corresponding to a trigger name, appended with ".time", when found in a correlation expression, is immediately replaced by the time abbreviation of the message that set the trigger, in 24-hour format ranging from 00:00:00 to 23:59:00. For example, the user can specify the comparison (12:00:?? in $mylatch.time).

Alert Formula Counter Names

These values identify the "Alert Formula" counter names, documented in Chapter 6 of this manual, which can be combined with math operators to create formulas that are subsequently alarmed via the "Alerts" screen. Each counter name is substituted by its counter value as part of the formula's evaluation.

Math operators include "+", "-", "*", "/", as well as selected functions such as "log10()", "log()", "exp()", "sqrt()", and other function supported by the standard C-math library. When necessary, parentheses should be used to specify the order of mathematical precedence.

Formulas can combine together various counters, creating a single result that is subsequently alarmed. One important application of alert formulas is to permit the user to assign weights to specific counters, such as in the following expression:

($severity/critical * 1) + ($severity/alert * 2) + $severity/emergency * 3)

The above formula returns a specific metric based upon the severity of messages of critical and higher, received during a user specified time interval.

$system/messages	This counter name is substituted with the total number of messages received since the CorreLog system started. This value corresponds to the message count appearing on the "Reports > Dashboard" screen.
$system/actions	This counter name is substituted with the total number of actions executed since the CorreLog system started. This value corresponds to the total of all counters appearing on the "Correlation > Actions" screen.
$system/triggers	This counter name is substituted with the total number of triggers that have been set since the CorreLog system started. This value corresponds to the total of counters appearing on the "Correlation > Triggers" screen.
$address/(ipaddr)	This counter name is substituted with the total number of messages from the specified IP address since the CorreLog system started. For example, the name "$address/10.1.2.1" is substituted with the total number of messages received from the 10.1.2.1 address. The value corresponds to the count for the IP address appearing on the "Messages > Devices" screen.

$facility/(facname)	This counter name is substituted with the total number of messages with the specified facility name since the CorreLog system started. For example, the name $facility/audit" is substituted with the total number of messages received with the "audit" facility specified. The value corresponds to the count for the facility appearing on the "Messages > Facilities" screen. The value of (facname) must be an official facility name, or a user defined facility.
$severity/(sevname)	This counter name is substituted with the total number of messages with the specified severity name since the CorreLog system started. For example, the name $severity/info" is substituted with the total number of messages received with a severity of "info". The value corresponds to the count for the severity appearing on the Messages > Severities" screen. The value of (sevname) must be an official severity name.
$trigger/(trigname)	This counter name is substituted with the total number of messages that have set the specified trigger name since the CorreLog system started. For example, the name $trigger/anymsg" is substituted with the total number of messages received that have triggered the "AnyMsg" trigger". The value corresponds to the count for the specified trigger appearing on the Correlation > Triggers" screen. The value of (trigname) must be an official trigger name.
$thread/(threaduid)	This counter name is substituted with the total number of messages that have been logged for the specified thread since the CorreLog system started. The value of (threaduid)" must be the numeric ID for the thread, viewable via the "Audit" link at the bottom of the Correlation > Threads" screen, or available via the "View Thread References" hyperlink on the Alert Formula Edit screen. For example, the name "$thread/000000010010 " is substituted with the total number of messages logged for the thread with UID 000000010010". The value corresponds to the count for the specified thread appearing on the Correlation >Threads" screen.

For Additional Help And Information…

Detailed specifications regarding the CorreLog Server, add-on components, and resources are available from our corporate website. Test software may be downloaded for immediate evaluation. Additionally, CorreLog is pleased to support proof-of-concepts, and provide technology proposals and demonstrations on request.

CorreLog, Inc., a privately held corporation, has produced software and framework components used successfully by hundreds of government and private operations worldwide. We deliver security information and event management (SIEM) software, combined with deep correlation functions, and advanced security solutions. CorreLog markets its solutions directly and through partners.

We are committed to advancing and redefining the state-of-art of system management, using open and standards-based protocols and methods. Visit our website today for more information.

CorreLog, Inc.
http://www.CorreLog.com
mailto:support@CorreLog.com

Alphabetical Index