UNIVERSAL ALERT SCRIPT (V 1.0)

Universal Alert Technology Add-On to Splunk®

Abstract

Send alerts via one or more industry standard methods including customized content and search results using one configurable alert script.

Burilliance, LLC of Virginia

Table of Contents

Introduction

The Universal Alert Script delivers Splunk® alerts to third party systems using various industry standard methods as well as via email and SMS through a single configurable script. Individual alerts are delivered based on the methods/destination configured in the alert (or default configuration values) by the individual designing the alert while delivering customized search result content. These capabilities make Splunk alerts more targeted, meaningful and useful thereby facilitating a more rapid resolution of the issue.

Current alerting methods supported include:

- SYSLOG
- EMAIL
- SMS
- OpsGenie®
- PagerDuty®

Additional alerting methods are under design consideration or may be added via a request to the developer.

This document provides all the documentation necessary to deploy and use the Universal Alert Script. The target audience is both administrators and alert designers.

Prerequisites

The Universal Alert Script is invoked through Splunk's standard alert (saved search) functionality. There are no Splunk specific pre-requisites other than standard Alert functionality available with a Splunk Enterprise license (it is not available in the free version) and the script has been tested against Splunk version 5 and 6.

The script is a python script and has been tested under python 2.75 (the current python interpreter distribution included with Splunk 6.0) on both windows and linux systems. There are certain python modules which are required and will be outlined in the installation and configuration section.

The script currently sends alerts to other systems using a variety of configurable mechanisms. Therefore, depending on how the administrator/user configures alerts to be sent, these other systems will need to be capable of receiving alerts in this format, and where applicable, interpret them properly. For example, a syslog server allowed to receive alerts from the Splunk server with an understanding of the key/value pairs, or an email account on an SMTP gateway.

The script has not been tested with other Splunk applications or technology add-ons, however as long as these use Splunk standard alert functionality, there is no reason the script should not function. There are certain minimum pre-requisite search result fields (key/value pairs) which are necessary for the script to function properly. If alerts are being generated without these key/value pairs present in the results, the alerts may have to be modified. These fields will be outlined in the subsequent sections.

Architecture

The Splunk knowledge management system is capable of generating alerts. These are referenced as "saved searches" in the vendor documentation. When an alert is generated "out of the box", the alert can be tracked in Splunk's alert manager and it can be sent via email, rss feed and/or be processed by a script. The results of the searches which generate alerts can be provided via the email notification, however these are not in an optimal format which can be systemically consumed. Numerous custom scripts for integrating alerts with specific third party systems have been developed, including those integrated into Splunk applications. Additionally, enterprises/end users have developed scripts customized to a given alert or groups of alerts. The strategy of a maintaining a custom script for each alert, or even groups of alerts does not scale well. These methods may suffice in certain enterprises where the volume is low and integration requirements are very discrete.

The Universal Alert Script was developed to address these challenges. Specifically, it is a single script which:

- sends alerts via any/all industry standard and/or 3rd party systems. Currently:
    - SYSLOG, EMAIL, SMS, OPSGENIE, PagerDuty
    - Other 3rd party systems with standard APIs can be easily integrated by the developer
- sends configurable/standardized content from alert arguments and/or search results
- allows the user to customize the content and destination(s) for each alert
- can be duplicated and use different configurations for each instance
- log all of it's activity so that alert generation was audit-able
- can run locally on the splunk search head or on another server to offload processing

The design concept of Universal Alert Script is that a generic alert mechanism can deliver alerts with the necessary content by using configuration file information, the standard Splunk alert script arguments and fields (key/value pairs) from the search results themselves.

Universal Alert Script consists of two files. These are the script file (.py) and the configuration (.conf) file. These files should be placed in $SPLUNK_HOME/bin/scripts. Under normal conditions, only one version of the script and configuration files are required. However, if an enterprise cannot agree on normally acceptable default key/value pairs and/or script argument values to make up alert content, multiple instances (using a different base name for the script and configuration file) can be deployed with different configurations for each. This will be discussed further under Installation, Configuration and Use Cases.

The configuration file corresponding to a given instance of the Universal Alert Script is a file with the same name, but with a .conf extension. For example the script may be called "uas.py" and the corresponding configuration file would be "uas.conf". As mentioned previously, the script and configuration files can be copied under any name.

One significant design objective/constraint of the Universal Alert Script is that a system receiving an alert from the script only requires one (1) result to generate the alert. For example, a Splunk alert may be based on a saved search that would retrieve multiple results (such as the CPU utilization across a number of different servers), yet only one of those results (e.g. one row of key/value pairs) is required to send an alert to an SNMP management system or a syslog system. Further, generally these types of

interfaces can only accept a single set of key value pairs. Therefore, when a saved search generates multiple results, only the most recent will be sent via the Universal Alert Script. Multiple alert sets are only required when investigating/troubleshooting the issue. The full result set can be easily retrieved through the search results URL which can be included in the notification. If it is necessary to provide multiple results in an initial notification by email (in addition to a SYSLOG message for instance), the standard Splunk email notification can also be used and include the results in the email notification.

Note: Normally, the Universal Alert Script retrieves the search results from the Splunk generated compressed CSV file. However, the results can also be retrieved from the Splunk search head using the Splunk REST API. This functionality was incorporated to allow the alert processing to be moved off of the Splunk search head to mitigate any performance considerations of the script on Splunk itself. Using this architecture is a special case and requires additional development to conform to specific enterprise requirements. Specifically, the "queuing of the script calls" to another system. The script will execute in about 50% of the time when using the CSV results rather than the REST API. Further, the performance impact on the Splunk search head when executing remotely and using the REST API will also have an impact on the Splunk search head as well. It is possible the REST API calls may impact the server performance as much or more than the script running locally.

Top Level Implementation Strategy

The implementation strategy of the Universal Alert Script is to define virtually all aspects for sending alerts via various methods in the configuration file. By doing so, items that will never change can be pre-configured such that any alert can use these items, albeit with alert specific information.

This is a straightforward concept for methods such as the SMTP email server and source email account for sending email notifications; or syslog server and port for sending syslog messages, etc. Items such as these can/should be pre-configured, and would likely not require any changes on a per alert basis or enterprise basis.

Less straightforward is message/notification content made up of key/value pairs from the search results or alert script arguments. Initially one might think that each group of support staff (e.g. system admins, storage, business analysis and network) may have entirely different key/value pairs which they would like to see as content in an alert, and they are correct. Yet, if one takes a higher level view, there is likely a common, normalized high level list of content which all alerts should contain. For instance, an alert management system might require the following list of items. The columns to the right show what elements these items could map to from different disciplines. These mappings from one specific field (as used in Splunk) can easily be mapped to these common, higher level names through the use of field aliases in Splunk.

High Level Field	**Network Field**	**SysAdmin Field**	**Web Server Field**
Component	Router(host) name	Server(host) name	WebServer(host) name
SubComponent	Interface	Disk (e.g. #1)	Transaction Type
State	Link State (Up/Down)	Utilization (e.g. 90%)	Transaction Status

Agreeing on a High Level Field naming standard is the first step in using the Universal Alert Script. Once this is accomplished, the individual disciplines can determine, on a field basis, which fields map to which higher level standard alert field. It is likely, and acceptable, that a subset of fields is used by a given group and/or alert. If a field is missing, the alert will show an empty value in a key/value pair. Using this agreement, the Universal Alert Script configuration file can be modified to include these common fields in the alert. Further, different key/value pairs are specified in the configuration file for each method of sending the alert. For instance, an SNMP trap may have very specific fields which must be mapped. Whereas an SMS message, which has to be shorter in length, may have fewer, shorter fields which are required. A syslog message may have to be less structured than an SNMP trap, consisting of key value pairs, some of which may be optional (empty). The Universal script accommodates all of these requirements.

There are circumstances where it may not be possible to reach common agreement on a high level field mapping between different support organizations, even though they may all be using the same alert management system. Universal Alert Script can accommodate this in two different ways. Each of these will be further elaborated on in the Use Case section.

Option 1 (Multiple Script Instances): This option applies when different support groups cannot agree on common high level fields, yet within each group agreement can be reached. In this case, multiple versions of the script and the associated configuration file can be deployed. For instance if the Network

and SysAdmin teams cannot agree on high level field mappings, but they can agree within each group, then two scripts with different configuration files can be deployed. For instance:

- uas_network.pl and uas_network.conf
- uas_sysadmin.pl and uas_sysadmin.conf

In this case, each team has their own script (albeit the script is identical) but the configuration files they depend on is different.

Option 2 (Custom Alert Fields): This option can be used on its own or in conjunction with option 1. This option allows the turning on of custom alert fields. This can be turned on in the configuration file for each notification method. (e.g. it can be allowed for email, but not other methods). This applies to all notification types currently implemented except for SNMP traps. The justification is that SNMP traps are highly structured and require a coordinated MIB definition file which cannot be different on an alert by alert basis.

Custom alert fields is just another key/value pair in the search results. The name of the field can be specified in the configuration file for the (each) script. The content of the results contained in a custom alert field is very structured. (It is generally the same as the field definitions in the conf file.) The results in this field must be a comma separated list of field name, descriptor, 'i' or 's' (for numeric or string data type). For example, the values in a custom alert field would look like the following:

> host,host name=,s,_time,eventtime=,s,interfacestate,status=,s,searchCount,# of events=,i

This field could be created within an alert with the following Splunk search syntax:

> … | eval
> alert_email_fields="host,hostname=,s,_time,eventtime=,s,interfacestate,status=,s,searchCount,
> # of events=,I"

The field names which are permitted in this list are either search result fields or any of the seven (7) Splunk alert arguments. These arguments have the following names (as defined in the script):

- searchCount Number of events returned
- searchTerms Search terms
- searchQuery Fully qualified query string
- searchName Name of saved search
- searchReason Reason saved search triggered
- searchURL URL/Permalink of saved search
- searchPath Path to raw saved results in Splunk instance (advanced)

The script will determine if custom fields are turned on for each specific alert_send method. If they are, and the key/value pair for custom defined fields exists in the results, it will use the custom defined fields. Otherwise it will default to using the fields defined in the configuration file.

Installation and Configuration

Python Requirements

A Python interpreter must be installed on the target system, which is part of a standard Splunk installation. The script is dependent on the following Python modules. All of these are installed with the standard Splunk installation.

- Sys
- Os
- Re
- Datetime
- Time
- ConfigParser
- Gzip
- Csv
- Urllib
- Urllib2
- Xml.dom's minidom
- Socket
- Smtplib

Script Installation

The Universal Alert Script is packaged like any other Splunk App. Specifically in a compressed tar.gz file with an spl extension. The initial part of the installation can be completed through the Splunk GUI from the Apps, Manage Apps dashboard by selecting "Install app from file". This will place the associated app files into the following directory:

<$SPLUNK_HOME>/etc/apps/uas referenced as ($APP_HOME)

This will create the following directories and files specific to the script, in addition to certain other default files. The directory structure conforms to Splunk's app directory/content guidelines:

<$APP_HOME>/bin
    /scripts/uas.py (the universal alert script)
<$APP_HOME>/default
    app.conf (general app configuration information, unrelated to script)
    uas.conf (baseline script configuration file)
<$APP_HOME>/default/data/ui/views
    uas_status.xml (a basic dashboard to provide the status of the uas script processing)
<$APP_HOME>/local
<$APP_HOME>/metadata

This directory structure is ONLY for distribution purposes and to support the app for the report view. The script and conf file will subsequently be moved.

Python Script Modification

The python script requires one modification, based on Splunk requirements. The first line of the script contains a reference to the python interpreter. In the distribution, this line is as follows, based on development being done on a windows installation.

#!C:\Program Files\Splunk\bin\python.exe

This line should be changed immediately after the installation to "#!<full path to python executable>.

After this change, this is the script version that should be used for any further copies of the script on your implementation.

Relocating/Renaming the Files

The extraction of the files into the uas application directory discussed above, is only for distribution purposes. These files should now be copied to one or more locations, depending on the usage requirements.

The script and configuration file should be ***copied*** to $SPLUNK_HOME/bin/scripts, which is the standard directory for all alert scripts run on Splunk. The two files should be renamed to a meaningful name for your enterprise (e.g. abc_uas.pl, and abc_uas.conf).

As mentioned previously, any number of copies of the script can be made in this directory with different names and corresponding configuration files.

Alert Triggering and Basic Requirements

The script is invoked by configuring a Splunk alert (saved search) and selecting Enable under Run a Script, and providing the script name (as defined above, including the .pl extension). Note: the other Splunk methods of notification (email and RSS) can still be used if desired.

In order to successfully invoke the script, the following result fields MUST be present or the script will fail:

- _time (this is Splunk's internal timestamp)
- _raw (this is Splunk's raw event data)

The _time and _raw fields are necessary to always include as mandatory fields. _time is used to timestamp the alert and _raw allows the script to ensure that the scripts own logging did not cause the alert, as this would create an infinite loop.

The user can ensure these fields will be provided in the search results by including the following in the search (please refer to the configuration section and use cases for further information about specifying fields to be included in search results):

… | fields + _time, raw

Additional search result fields will be required based on the specific use case. These will be covered configuration section below.

Splunk Alert Configuration

Saved searches configured for alerts which use UAS must meet a few basic configuration requirements. Specifically, the following parameters/fields must be set correctly, each of which is highlighted on the sample Searches and Reports edit screen:

- Schedule this search; must be selected for the alert to trigger
- Expiration; This field must be set to some period of time. The recommendation is 24 hours. However, it can be set as short as 15 minutes.
- Run a script Enable; must be selected for the script to run
- File name of Shell Script to run; must be the name of the script
- List in Triggered Alerts; must be selected in order to retain the results so they are able to be retrieved.

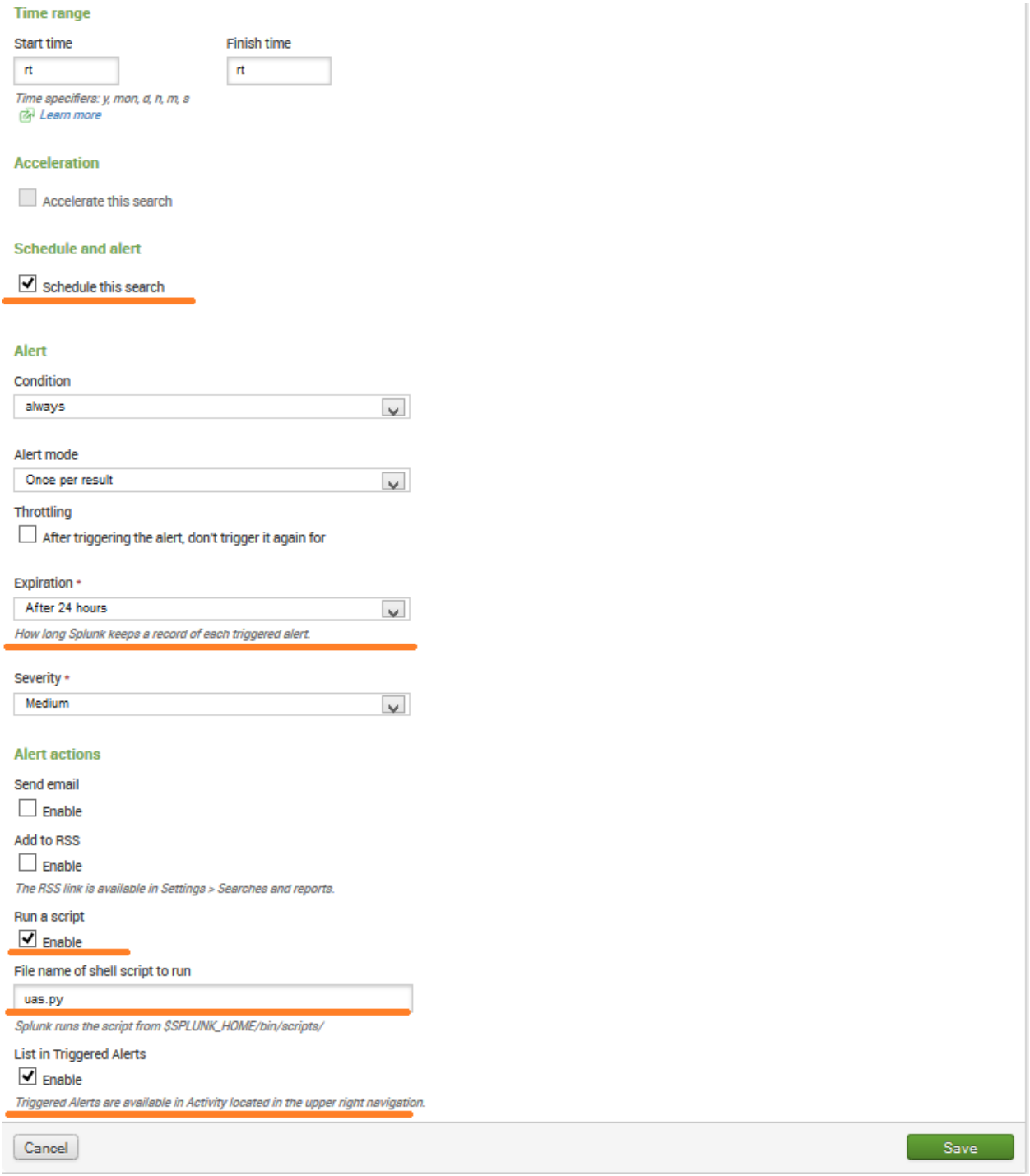

Figure 1: Splunk Alert Configuration

Configuration

Security Considerations

The configuration files for the script contain some sensitive information. This includes items like user name and password for the sending email account, host name and ports for syslog and SNMP, customer keys for OpsGenie and PagerDuty. These items are stored in clear text. At this time there are no plans for encrypting these items as there is very little which could be gained from compromises to these items. Specifically, while this information is somewhat sensitive, a compromise would only compromise an administrative send only email account or a customerkey for alerting via OpsGenie and PagerDuty. Further, these files should be stored in a directory whose permissions are only for the Splunk system and root, which makes obtaining the information in these files difficult. Lastly, without incorporating some sort of public key encryption method, which would slow down the script processing, there is little value in any local encryption as access to the configuration file would also mean the script was accessed and any local encryption could be therefore deduced.

Configuration File Structure and Items

The configuration file for a given instance of the script is a standard .conf formatted file. It consists of stanzas, and within each stanza there are configuration variables. Stanzas are identified by a name in square brackets. Configuration variables are a name followed by an equal sign and a value. The Universal Alert Script configuration file consists of one stanza for the script in general and a stanza for each of the alert methods. Comments in the configuration file are any line which begins with a hashtag (#) symbol. A simplified version of the configuration file would look like:

```
# Script configuration
[script]
varable1 = “test”
#SNMP configuration
[snmp]
varable1 = “test”
#EMAIL configuration
[email]
variable2 = “test2”
```

A complete copy of the Universal Alert Script configuration file with full comments is included in the References section of this document. The configuration file controls all aspects of the operation of the script and each type of alert sending method.

The configuration file template provided with the software distribution and its comments fully document the purpose of each configuration item. However, the following sections will overview the script stanza, general items included in each of the alert method stanzas and some of the more important configuration items.

In these descriptions, the standard naming for a configuration variable stanzaname.variablename is used. Therefore email.host would refer to an item in:

```
[email]
host=”my.smtpserver.com”
```

Script Control Configuration Items

The configuration file stanza [script] contains configuration items for how the script will operate overall, the level of logging done by the script and how to log the script results.

The script.mandatory_fields item indicates which fields the script must have in the results file. As mentioned previously, this includes _time and _raw. Additional mandatory fields can be included to ensure that the search results for an alert ALWAYS has these fields if it is to be processed. Any alert which does not have these fields will be logged indicating missing fields.

The script.time and script.raw variables specify the field names of these two keys, and should not be changed.

The script.alertsend variable (field name default of alert_send, but can be renamed) determines what methods to use to send the alert. The field result, if present, must contain a comma separated (if more than one value is used) list of acceptable values.

If the alert_send field is not present in the search results, the value specified in the conf file for script.alertsend_default, if defined, will be used. Finally, the script.alertsend_default_always variable in the configuration file, if set to true (1), will force the default send method to always be sent, in addition to those specified in the alert_send field. The currently acceptable values for these fields are:

- SMTP
- SYSLOG
- EMAIL
- SMS
- OPSGENIE
- PAGERDUTY

By example, for sending both a SYSLOG message and an EMAIL message, the search would include the following:

… | eval alert_send=”SYSLOG,EMAIL”

script.process_alerts controls what alert methods the script will process and should only be modified by the script developer.

script.max_rows indicates how many rows of search results will be read in and processed for any given alert. max_rows should be left at one (1) as of this writing since the script does not do anything with multiple results at this time other than allow the inclusion of a count of search results in the message. The rational for this is that an alert is due to a condition being met. That condition can be described/summarized by including a single set of key/value pairs in the notification. While resolving an issue may require further investigation into all the results making up the alert, these can be obtained systemically after the fact either by human or systemic intervention. Lastly, if desired, the script can be configured to include the URL for the search results as part of the notification making obtaining the full results straightforward, when necessary. Meanwhile, alerts should be small and concise. Lastly, if an enterprise desires to get all the results sent via email notification, the standard Splunk email notification method can be used in lieu of emails generated by the script.

The results_from variable controls whether the script obtains its results from either the CSV file (on the local Splunk search head) or the REST API. If the script is running locally on the Splunk search head, this should be CSV. If the script is run on a remote server (this is an advanced topic) then REST should be used. If REST is used, there are a number of variables specific to it as follows (these can be ignored if the REST API is not used):

- splunk_uid: The user ID used for logging into Splunk
- splunk_pwd: The password used for logging into Splunk.
- splunk_rest_port: The TCP port used for the Splunk Rest API.
- splunk_auth_url: The URL for Splunk REST API authorization.
- splunk_search_url: The URL for Splunk REST API search.
- splunk_ssl_verify_mode: 0 to not perform SSL verification.
- splunk_ssl_verify_hostname: 0 to not verify the Splunk hostname.

script.quiet_timeXX, where XX is 01 through as many rows as necessary. This (these items) specifies the master quiet time such that NO alerts will be processed, nor will they be queued, so they are essentially suppressed.

script.override_quiet specifies a field name which, if processing an alert and this key/value pair exists in the search results, and the value is set to one (1), then the alert will be processed, regardless if quiet time is active, at either the master quiet time or individual alert quiet time level.

script.log_level specifies the amount of logging the script will do. Zero (0) is no logging, one (1) is script errors only, and two (2) is all, including alert delivery success or failure of each alert method. If the value is set to 1, a log will be sent if one or more of the alerts fail in some way.

The remaining fields in the script stanza specify the syslog parameters for how to log the script activity using standard syslog. Generally, this is expected to be a remote server. However, the local server address (127.0.0.1) can be used. These parameters should be familiar to someone familiar with syslog functionality, and are further documented in the config file.

Two other fields exist for syslog, and similar fields are used in other stanzas as well.

> script.sysl_linesep specifies the line separator between key value pairs. For syslog, this is typically a “, “ as there is no new line character used in syslogging.
>
> script.sysl_text_del specifies what delimiter should be used to surround string values.

Lastly, also associated with syslog, is script.fieldXX. (Please note, this construct is used in other stanzas for each of the alert methods as well to define the content of the alerts.) These are named field01…fieldXX, where XX are digits. Further, these should be left in numerical order. These variables are used to specify, in this case, the information fields that will be used for syslogging the script results. In other contexts (e.g. the email stanza) these variable names are used to specify the information fields put in the email message. The structure of these variables is:

> fieldXX = field or script argument name, descriptor, ‘i’ or ‘s’ (for numeric or string)

Therefore, in the originally distributed version of the script configuration file, the following lines exist indicating that 7 key/value pairs will be included in the script syslog, as follows. In this case, the syslog

will output all of the script argument values with a descriptor which is the name followed by equals (=), and that all values are string, with the exception of the searchCount value. The string values will have the value surrounded by a quote (“) if that is what was specified in the sysl_text_del configuration variable. While both script arguments and search result key/value pairs could be included in the script syslog information, it is expected that only the script arguments would be used here.

field01 = searchCount, searchCount=, i
field02 = searchTerms, searchTerms=, s
field03 = searchQuery, searchQuery=, s
field04 = searchName, searchName=, s
field05 = searchReason, searchReason=, s
field06 = searchURL, searchURL=, s
field07 = searchPath, searchPath=, s

The script will also output a scriptname, scriptstatus and scriptdetail key/value pairs in the syslog message by default, in addition to those fields configured above. scriptstatus will indicated either “Success” or “Failure”. scriptdetail will indicate the success or failure of each alert method, or the reason for a script failure.

There are a few other important notes regarding the script functionality.

New Lines and/or Carriage Returns have been known to be included in the value of the key/value pair in the results of a search. These are known to cause problems when used to output in various alert methods. (e.g. a syslog server will stop processing all characters after a new line; an email body will be reformatted with the additional newline.) Therefore, the script will accept these newline characters but each will be replaced with a space.

It is possible that double quotes (“) can be embedded in the search arguments and the value of a key/value pair in the results of a search. These will also cause problems in the output to various alert methods. Therefore, the script will accept these double quotes, but each will be replaced with a single quote (‘).

Alert Method Configuration Items

Each alert method has its own stanza in the configuration file, and determines how each alert method will be processed. This section is an overview of the configuration for all the methods, however each has some unique variables defined based on the requirement of the method.

Each alert method stanza begins with the enabled configuration variable. If this is set to 1, the script will process that alert method, otherwise, it will not.

Similar to the script stanza, each alert has one or more quiet_timeXX values, which specifies the quiet time for the individual alert method.

Each alert method has an allow_custom_fields configuration variable. This is either false (0) custom fields are not allowed, or true (1) custom fields are allowed. This variable exists for all alert types, but should NOT be set to true (1) for SNMP as SNMP has a custom MIB definition. Other alert types are at the administrator’s discretion, but care should be taken to ensure this will not cause system compatibility issues. For instance, if custom fields are used for SYSLOG alert methods, will the system

interpreting those key/value pairs understand this. For email and SMS, this should not be an issue if it is for human consumption, unless other systems are interpreting these messages.

With the exception of SNMP, each of the other alert method stanza's have a customfields variable. If the allow_custom_fields is true for a given alert method, the alert designer may specify which fields (key/value pairs) will be used in the "body" and "subject" of an alert. This custom field specification, if it exists, will override the fields specified in the configuration file. Specifically, the mapping is as follows:

Normally Used Fields from conf File	**Custom Field Specification**
alltypes.fieldXX = field,descriptor,'i' or 's'	*alltypes*.customfields = (one of) alert_syslog_fields alert_email_fields alert_sms_fields alert_opsgenie_fields alert_pagerduty_fields
email.subjectfield = normal_subject_field	email.customsubject = alert_email_subject
sms.subjectfield = normal_subject_field	sms.customsubject = alert_sms_subject
opsgenie.message = normal_message_field	opsgenie.custommessage = alert_opsgenie_message
pagerduty.message = normal_message_field	pagerduty.custommessage = alert_pagerduty_message

Therefore, if the custom field named in the configuration file (e.g. alert_syslog_fields) exists in the search results and contains a list of field name, descriptor and type, these will be used vs. the normal configuration variables (fieldXX) for syslog messages. The same applies for the subject fields, however the values in these result fields are a singular string of field, descriptor, 'i' or 's'. (e.g. searchName,TheSearchName=,s).

EMAIL and SMS alert method's stanzas are almost identical (since SMS is sent via an email gateway) and contain a few other user related fields. These are:

subject; is the Subject field "leader" (e.g. it may be set to "Alert from Splunk: "), and the value of the subject field would be appended to it. In the case of either email or SMS, this can be left blank, and with SMS, it would normally be left blank.

subjectfield; is the "standard" subject field name who's value is to be used if no customsubject is defined. Again, this would likely be left blank in the SMS alert type.

SMS alert also has a specific configuration value called maxlength. This sets the maximum amount of characters the SMS message can contain. SMS messages are constructed, including a subject line, if applicable. If the length of the message, including the descriptor, values and any delimiters or line separators exceeds this length, the total message will be truncated to maxlength.

The remaining fields in all of the stanzas of the configuration file are specific to the type of alert method (e.g. host for SMTP server, port number, support for TLS etc.) and should be self-explanatory to those familiar with these technologies and are well commented in the configuration file.

Each stanza also has the fieldXX configuration variables which defines the “standard” content in key/value pairs (in lieu of customfields) which will be provided in the alert. This field in the OpsGenie and PagerDuty stanzas is the information which is included in the “details” portion of the respective REST API interface, documented on their website. Additional uses cases and possible inclusion of result fields in some of the OpsGenie specific API fields is under consideration.

For email, sms, and OpsGenie the conf file specifies which field should be used for the destination. However, in the event these fields don’t exist in the search results, a default value (not field name) can be used, if specified in the conf file. The table below shows these variables and their relationships.

Normal Field Variable	**Configuration Default Variable**	**Type of Default Variable**
email.sendtofield=alert_email_to	email.sendtofield_default	Value
sms.sendtofield=alert_sms_to	sms.sendtofield_default	Value
opsgenie.recipients=alert_opsgenie_to	opsgenie.recipients_default	Value

Script Testing

Basic Testing

Prior to configuring any real life alerts to use the Universal Alert Script, it is advised to test it first using the following methodology. This ensures the basic functionality (e.g. reaching the SNMP, SYSLOG, EMAIL, OpsGenie and/or PagerDuty servers) is working. The testing should be performed test each alert method which is going to be used at a time.

The first step is to modify the configuration file to conform to your use requirements. This version of the script and associated configuration file can then be placed in any directory on your system for testing.

The second step is to obtain a results file from any existing search/alert. This can also be placed in the same test directory. Further, while this is a compressed file, you may decompress the file and use the native CSV file. The script is set up to look for the file extension (.gz for compressed, and .csv for uncompressed native CSV) in the argument passed to it and then use either. This was done specifically to allow users to manually manipulate a CSV file for testing without having to decompress and compress each time.

This results file must contain, at a minimum, the _time and _raw fields, which most search results will contain by default.

The results file will also contain some other fields, likely NOT those you would eventually require for alerts. However, the conf file can be modified to just use whatever fields are necessary to populate some test results out to the target alert method. Since alert_send will not be in the results, the default should be set, likely to SYSLOG, at least to start, and then move on to email and others.

The testing can be initiated by writing a script (batch on Windows, shell on Linux) which calls the alert script with the appropriate parameters for testing. The ONLY limitations of this testing scenario is it does not allow using REST (the Splunk REST API) to obtain the search results and the parameter passed to the script for searchURL will be a dummy value and therefore not functional.

The script would look like as follows, where <$TEST_DIR> is the test directory where the files are stored.

```
python <$TEST_DIR>\uas.py 1 "dummy search terms" "dummy fully qualified query string"
"dummysearchname" "dummy description or trigger reason" "http://dummy.url" ""
"<$TEST_DIR>\tmp_0.csv.gz"
```

These 8 arguments are the same 8 that Splunk would pass to the alert script. In the above, the (1) immediately following the uas.pl is the searchCount argument. The file name at the end “tmp_0.csv.gz” is the results file previously mentioned. Again, this could also have been “tmp_0.csv” if you want to manipulate the CSV file directly. And to re-enforce, after dummy.url, that IS a blank argument, one which was deprecated by Splunk, but is still a placeholder for backward compatibility.

After running the script, you may then look at the scripts SYSLOG data (likely within Splunk) by searching on the name of the Universal Alert Script. This should show at least the script syslog information identifying the success or failure. It may also show the results of the SYSLOG method that was used to send an alert, if this was requested. The other test results would be to determine if alerts were received by email, SMS, OpsGenie, or PagerDuty.

Additional testing can be performed by manipulating the content of the CSV file, and running the tests again. This might include adding an alert_send field and populating it with another method, for instance. Other fields planned for the ultimate default fields to be used by your organization could also be populated, manipulations to the conf file to include those fields in the various alert methods and the tests run to generate sample notifications conforming to these defaults.

Advanced Testing (REST API)

An organization which plans to use the Universal Alert Script but run the script on a system other than the Splunk search head would want to perform additional tests, and is a much more advanced topic which should likely involve an engagement with the Universal Alert Script developer.

As mentioned previously, the default behavior of retrieving the search results locally out of the compressed CSV file can be changed to use the REST API. If this is planned to be used, this can also be tested on the single search head. In this case the test batch/shell script would have to be modified to include the URL to a set of actual, actively store by Splunk, search results. Then, by changing the conf file variable script.results_from to REST, the script will attempt to retrieve the results using the REST API.

The implementation of the script on a different/remote server will require additional integration steps which do not have a standard setup at this time. These steps will require that a method of triggering the script be established. This could involve using the REST API to obtain active alerts on the search head, or it could involve using a very basic script to queue the work need to be done to another system. In either case, some capability on the remote system would have to monitor the need to process alerts and call the Universal Alert Script to do so. This call would have to pass the same arguments to the script which Splunk uses. The Universal Alert Script CAN also receive a 9th argument. This argument, if present, can include a configuration name. Splunk cannot natively do this. However, if a remote server was being used, a single version of the script can be used while allowing the method which queues work to the script to include this configuration name. This name would correspond to a filename containing the appropriate configuration file.

Again, these are advanced topics which should be discussed with the developer prior to attempting to implement them.

Script Performance Views

UAS syslogs its own activity. The level of logging is controlled by the conf file. Logging levels are defined as:

- 0 – No Logging
- 1 – Log Failures only
- 2 – Log all activity

These logs can provide views into if the script is succeeding or failing, and when it fails, the reason why. This information is invaluable during the initial setup and testing of UAS. Additionally, in the event a failure occurs and the alert script did not succeed, the logs provide information as to why.

The UAS Splunk app will appear in the Splunk application views based on the access levels defined during installation. There is one dashboard defined in the app. This dashboard is show below. The dashboard shows:

- All activity in the top pane of successes and failures
- The Failures by type of failure in the lower left pane
- The Failures by script name in the lower right pane.

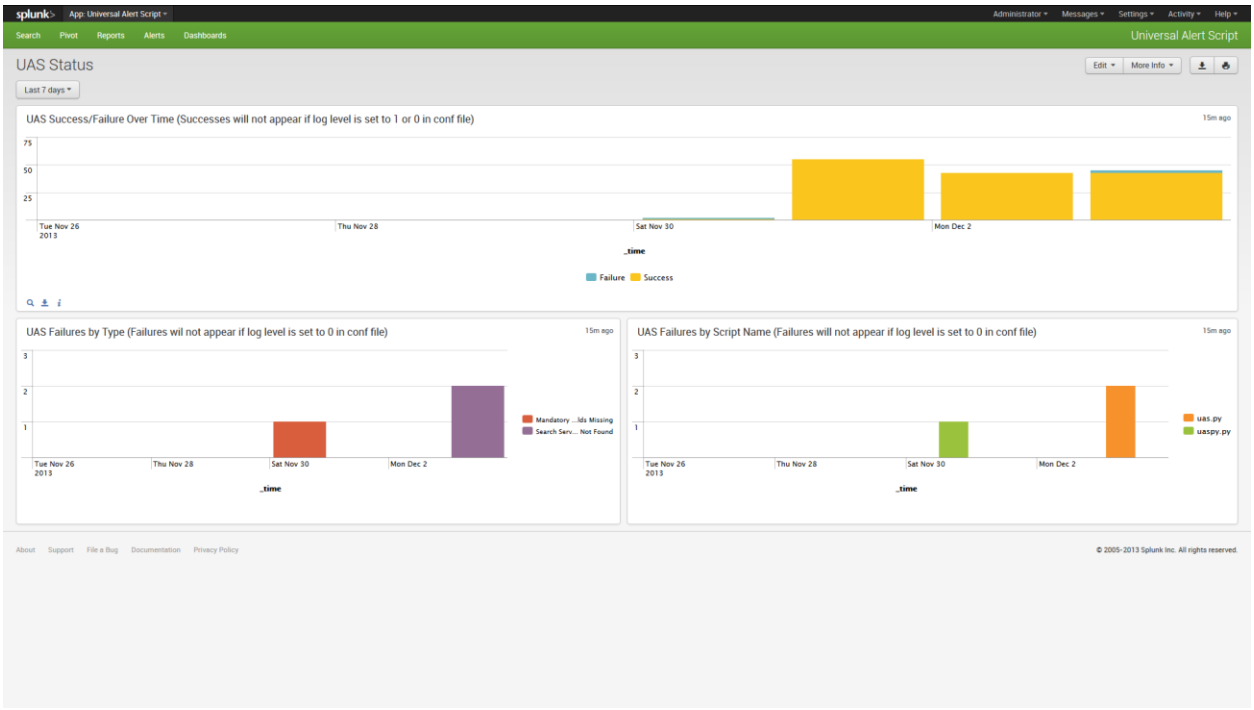

Figure 2: UAS Performance Dashboard

Use Cases

The following address three use cases for the Universal Alert Script. In addition to describing the deployment and basic configuration, use of Splunk macros to standardize and simplify the creation of alerts and their input to the script will be described.

Use Case 1 – Enterprise Wide Alert Standards

This use case addresses the situation where an entire enterprise can agree on standards for what key/value pairs (content) will be included in any given alert method. The corporate standard will include alerts being sent to a syslog server and optionally, alert designers can send SMS messages to user groups.

In this case, a single copy of the script file will be deployed called alert_gen_process.pl with a corresponding alert_gen_process.conf file into the (splunk home directory)/bin/scripts directory.

The alert_gen_process.conf file will be updated to specify that:

script.allowed_alerts = “SYSLOG,SMS”

Various other configuration file items will be updated to specify the parameters for the SYSLOG server communications and the sending email server and account for the SMS messages.

Since we have agreed on standard fields across the enterprise, the fields for the syslog message will be updated in the configuration file’s syslog stanza as follows:

field01 = _time, event_time=, s
field02 = alert_description, description=, s
field03 = host, hostname=, s
field04 = alert_subcomponent, subcomponent=, s
field05 = searchCount, Numbofevents=, i
field06 = searchName, SearchName=, s
field07 = alert_send, SendList=, s

The fields for the SMS message will be updated in the configuration file’s SMS stanza as follows:

field01 = host, hostname=, s
field02 = alert_subcomponent, subcomponent=, s
field03 = alert_description, description=, s

We have not updated the text delimiter or line separator fields from the standard configuration file.

Since the enterprise has agreed to standard fields to be included in alert methods, a couple of Splunk search macros are created to ensure that the appropriate fields are included. The two search macros are as follows.

alert_syslog_only(arg1) = fields + _time, _raw, host, alert_subcomponent | eval alert_send=" SYSLOG" | eval alert_description="$arg1$"

alert_syslog_and_sms(arg1) = fields + _time, _raw, host, alert_subcomponent, alert_sms_to | eval alert_send=" SYSLOG,SMS" | eval alert_description="$arg1$"

Notes:

- alert_send and alert_description are not included in the field + clause since these fields are explicitly defined in the search, they are included by default.
- There are fields defined for the syslog message which are script arguments (e.g. searchName, searchCount), therefore these are also not called out in the fields + clause.
- The splunk macro has a parameter which is used to supply the field value for alert_description. When this macro is used in an alert search, the search will pass a human readable description of the problem to the macro which will ensure the alert_description field is populated accordingly.

Now an alert (saved search) can be configured to call alert_gen_process.pl. The search for this would consist of some custom search to identify the alert condition (which will not be specified here) but suppose this is an alert for a network interface down. The search would look like:

… | `alert_syslog_only("Network Interface Down")`

This would result in a syslog message being sent to the syslog server like the following:

event_time="10-15-2013 21:14:30.000 -0000", description="Network Interface Down", hostname="192.168.1.193", subcomponent="eth2", Numbofevents=1, SearchName="Test", SendList=" SYSLOG"

If the alert designer wished to send this alert to SMS as well, the search would have been modified as follows with the corresponding SMS message looking like this:

… | eval alert_sms_to="NOC@abc.com" | `alert_syslog_and_sms("Network Interface Down")`

host=192.168.1.193
subcomponent=eth2
alert_description=Network Interface Down

Since the macro has alert_send set to "SYSLOG,SMS", a syslog message would also be generated, just as it was for the previous description.

The use of the alert_email_to, alert_sms_to fields, as in the example above, is an email address. Although multiple email addresses could be provided in a comma separated list, the best practice would be for these to be email group addresses and manage the members of the group on the email system.

Use Case 2 – Alert Standards within Specific Groups

This use case addresses the situation where an enterprise can NOT agree on standards for what key/value pairs (content) will be included in any given alert method and needs different standards for various support groups. In this case the sysadmin group wants both SYSLOG and SMS capabilities whereas the network group wants SYSLOG and EMAIL capabilities.

In this case, two copies of the script file will be deployed with a corresponding configuration files into the (splunk home directory)/bin/scripts directory.

The two scripts might be as follows and there would be corresponding .conf files:

alert_sysadmin.pl,
alert_network.pl

The two configuration files will be updated to specify that:

The sysadmin script configuration file:

script.allowed_alerts = “SYSLOG,SMS”
The rest of the sysadmin configuration file might remain as it was in the Case 1 example.

The network script configuration file:

script.allowed_alerts = “SYSLOG,EMAIL”
mandatory_fields = _time, _raw, alert_send, router_name, interface

The fieldXX specifications for both syslog and email might be modified as follows:

email.field01 = router_name, Router Name=, s
email.field02 = interface, Interface=, s

email.subject = “Network Alert From Splunk: “
email.subjectfield = alert_description

The sysadmin group would use a search macro that might look something like:

alert_sysadmin_syslog_and_sms(arg1) = fields + _time, _raw, host, alert_subcomponent, source, sourcetype | eval alert_send=" SYSLOG,SMS" | eval alert_description="$arg1$"

Meanwhile, the network group would use a search macro that might look something like:

alert_network_syslog_and_email(arg1) = fields + _time, _raw , routername, interface | eval alert_send=" SYSLOG,EMAIL" | eval alert_description="$arg1$"

The associated searches for the sysadmin group would be:

…| eval alert_sms_to=”sysadmins@abc.com” | `alert_sysadmin_syslog_and_sms(“Disk Full”)`

This would return an SMS message for the sysadmin group which would look like:

host=192.168.1.193
subcomponent=Disk1
alert_description=Disk Full

The associated searches for the network group would be:

…| eval alert_email_to=”networksupport@abc.com” | `alert_network_syslog_and_email(“Interface Down”)`

This would return an email message for the network support group which would look like:

> Subject: Network Alert From Splunk: Interface Down
> Router Name=SomeRouterName
> Interface=SomeInterfaceName

Use Case 3 – Custom Alert Fields

This use case addresses the situation where an entire enterprise can agree on standards for what key/value pairs (content) will be included in any given alert method, but wishes to allow custom alert fields for EMAIL and SMS alerts to be optionally specified by the alert designer.

The following description is for a network team that is generating an alert but due to the complexities of the alert, they are using custom fields for the email message which is generated and want to include host, routing protocol and state

In this case, the configuration file would have:

> email.allow_custom_fields = 1
> email.customfields = alert_email_fields

In this case the search macro which would be used is:

alert_syslog_and_email(arg1) = | eval alert_send=" SYSLOG,EMAIL" | eval alert_description="$arg1$"

Very Important Note: Splunk searches can only support 1 “fields +” clause in a search, whether this is within the primary search or within a search macro. Therefore, since custom fields are being defined for inclusion in the email, and the fields we are specifying may not be explicitly defined in the search, a field + clause in the primary search is required. Further, since we can’t have two clauses, and the field + clause was eliminated in the macro above, ALL of the fields must be specified in the primary search. It will be up to the field designer to add a fields + clause to the primary search to ensure they are captured in the results, which is shown below, before the macro call.

The search which would use this would look like:

… | eval alert_email_fields=”host,Host=,s,route_protocol,Protocol=,s,state,Protocol State=,s” | fields + _time, _raw, route_protocol, state | `alert_syslog_and_email(“Routing Protocol State Change”)`

In this case, the resulting email would like like:

> Subject: Network Alert From Splunk: Routing Protocol State Change
> Host= SomeRouterName
> Protocol=SomeRoutingProtocol
> State=SomeRoutingProtocolState

Configuration File Important Notes

All items referenced in the configuration file by the script are case sensitive, so changing the case of the existing variable names should NEVER occur.

If field(s) are identified to be used in an alert, and these fields don't exist in the search results, the descriptor will be displayed, but with a null value.

If the configuration file does not exist for a given named script, the script will fail without any logging.

Logging done by the script as configured in the configuration file is an important tool when trouble shooting an issue with alert generation as this contains all the script arguments (including the location of the results file) and the status of the script processing. A static key called scriptstatus will indicate either "Success" or "Failure". Failure indicates the entire script failed (for instance if the results file could not be opened). Another static key called scriptdetail will provide more information. If the status was Failure, this will indicated the reason for failure. If the script is successful, scriptdetail will indicate either a +/- followed by the alert method, for each alert method processed. A + indicates the alert method succeeded. A – indicates it failed and a reason code will follow in parentheses.

Performance Considerations

Universal Alert Script will execute for each and every Splunk alert which is triggered. As a result, consideration must be given to ensur the performance of the Splunk system as a whole, and the alert processing done by the script. Specifically, that sufficient resources exist to ensure all alerts are processed in a timely fashion and the processing of these does not affect the Splunk system. The following guidelines are specific to UAS and do not address overall tuning of the Splunk system as a whole.

The script(s), associated configuration file(s), python modules and python interpreter must be read into memory for execution for each alert which is triggered. Therefore, these items should be placed into high speed, low latency storage in order to ensure the lowest possible read times for these items.

Concurrent alerts and/or alert floods could occur based on events being process by Splunk coming from the monitored infrastructure. Each alert will spawn another process. If an alert flood occurs, the number of processes and memory limitations of the host system could be exceeded. If this occurs, the performance of the host system could be impacted. Care should be taken to ensure the system has the required resources for the anticipated concurrency and appropriate protections should be put in place to ensure all available system resources do not run out. Perhaps most importantly, care should be taken in the design of the saved searches/alerts to ensure appropriate throttling is done so as not to overwhelm the host system or the receiving alert management targets. It should be noted that the script has a built in capability to ensure poorly designed alerts do not trigger based on syslog events coming from the script itself. This will eliminate the potential for an infinite loop caused by the alerting script. Therefore, it is important that monitoring of the alert script should not be processed by the alert script, or these will fail, without logging.

The speed of execution of the script is mostly dependent on its interaction with receiving systems. For instance, the speed of response from the SMTP server for email, the web hosted OpsGenie and PagerDuty system or the SNMP listener will dictate how quickly the script completes its tasks. Optimizing the latency to/from these systems and the response time of these systems will have a dramatic effect on the script performance.

The configuration file can be made significantly smaller by eliminating all of the comment lines which are only there for human understanding of the file. It is recommended that all comment lines be eliminated in the production version of the configuration file.

A more advance topic would be editing of the configuration and script file to eliminate unnecessary stanzas from the configuration file and elimination of unnecessary subroutines in the script. For instance, if there is never any intention to use SNMP, this subroutine and the associated configuration stanza could be eliminated. While straightforward, this is not something which should be done without complete understanding of the script and configuration file. Further, the script will error out if an alert type is allowed, and configured by the alert or by default, and the subroutine or associated configuration variables are missing.

Lastly, consultation with Splunk engineering or professional services regarding the system architecture for the Splunk deployment should take place to ensure optimal performance. If the alerting capabilities are mission critical, the system performing alert searches and executing the script should likely be dedicated to this purpose so that user driven searches do not impact the performance of this system.

Upgrade Instructions

Since this is the first release of the Universal Alert Script, there are no upgraded requirements. However, in the future, the requirement will generally be to ensure the configuration file and the alert script are from the same release so that all elements are defined.