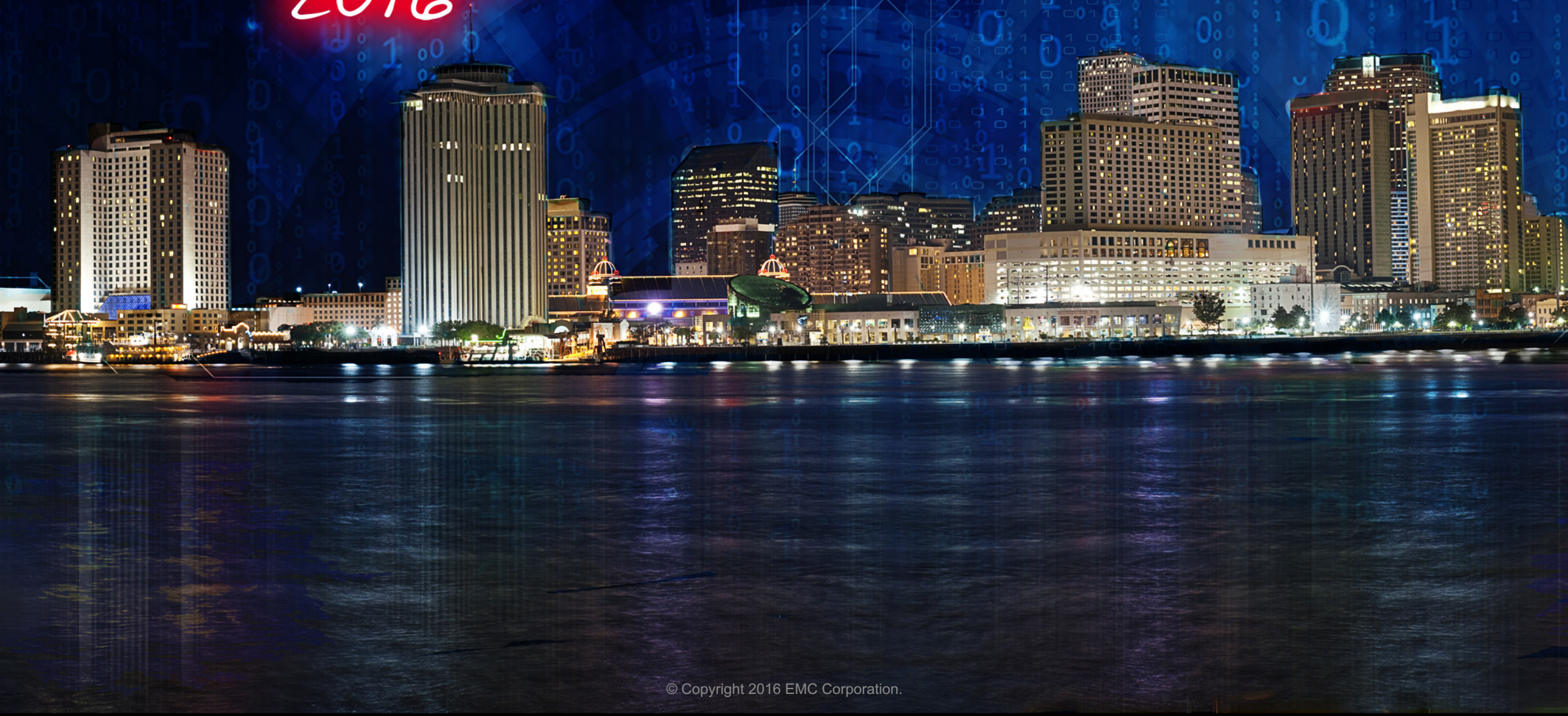


# RSA® Charge 2016





# Log Parsers – Soup To Nuts

---

Joe Gumke, @joegumke

joe.gumke@usbank.com

# Presentation Overview

- How To Consume Event Sources
- How To Normalize
- How To Deploy
- How To Manage Log Parsers
- How To Troubleshoot
- How To Validate
- Tips

# How To Consume Event Sources

- What do you want to do with logs? (WHY)
  - Operational
  - Security
  - Policy / regulatory(external/internal)
- Why do you need this log source?
- What specifically in the logs do you need?
  - Gather Use Cases to justify
  - Document **ANYTHING** And **EVERYTHING** about your log sources,format,etc...
  - Will come in handy when building your own parser

# How To Consume Event Sources

















System Origination	Contact	SA infrastructure	Log Source Details
Source Hostname(s)	Custom Parser Author	Decoders	log format of the log source
Source IP(s)	Estimated Log Volume Per Sec/Hr/Day	Concentrators	Any supporting documentation for log format
Log Transport Method	Use Cases	Collectors	Log Format Description
Application/System Version	System Contact	Parser Used by this log source (device.type)	SA Field Mappings
Test/Dev Systems	RSA Parser Author	Example SA Query	
PROD Systems			

# How To Consume Event Sources

- How to consume events within the SIEM

- What does product/system support ?
- What does RSA support?
  - ODBC? SYSLOG? File?

A

Event Source Name	Version	Parser Name	Collection Method	Instructions
Accurev	6.0.1	accurev	File	  <a href="#">Additional Downloads</a> 
Actiance Vantage	12.2	actiancevantage	ODBC	 
ActivIdentity 4TRES S AAA Server	6.4.1	actividentity	ODBC	 
AirMagnet Enterprise	7.5, 8.5, 10.1	airmagnetenterprise	Syslog	 
AirTight Management Console	7.0, 7.1 U4	airtightmc	Syslog	 
Alcatel-Lucent OmniSwitch	6600, 6850, 9700	alcatelomniSwitch	Syslog, SNMP	 
Apache HTTP Server	2.1, 2.2, 2.4	apache	Syslog, File	  <a href="#">Additional Downloads</a> 

- Ask Account rep / submit ticket to support if needed, otherwise build your own

- **RSA Supported Event Sources**

- [http://sadocs.emc.com/0\\_en-us/300\\_RSA\\_ContentAndResources/03\\_Supported\\_Event\\_Sources](http://sadocs.emc.com/0_en-us/300_RSA_ContentAndResources/03_Supported_Event_Sources)

# How To Consume Event Sources

- File Type Specifications
  - modifies message when its read into NetWitness Logs & Packets
  - **Log collector Path:** /etc/netwitness/ng/logcollection/content/collection/file/
- Needed for file/odbc based event transmissions
  - prepends characters to raw logs ingested into SYSTEM > “%--4”
- MAKE SURE syntax is correct on filetypepec file...
  - otherwise your log collector will break, not load.
  - **Resource :** [https://sadoes.emc.com/0\\_en-us/089\\_105InfCtr/135\\_LCGds/40\\_FileProto/10\\_FileProc/00\\_CnfgFileESRec/20\\_FileTypspec](https://sadoes.emc.com/0_en-us/089_105InfCtr/135_LCGds/40_FileProto/10_FileProc/00_CnfgFileESRec/20_FileTypspec)



# How to troubleshoot - Log Collector File Type Spec

## Issue:

Modifying a log collector configuration that breaks the log collector, which wipes the log source configurations from the log collector.

## Resolution:

NetWitness Logs & Packets creates a save state on each log collector that contains all log source configurations.

**Location :** [/etc/netwitness/ng/NwLogcollector.cfg](#)

Every time the log collector does not load properly, it will create a new "NwLogcollector.cfg",

and create a new file with iteration of .1..

for instance, test collector shows this :

If for some reason the configuration is broke, and wipes information.

The newly created configuration should be named : NwLogcollector.cfg.1

## To resolve this, do the following :

1. find the last 'good' state of the configuration that was setup.
2. stop nwlogcollector
3. copy the old 'good' configuration and rename (overwrite) existing NwLogcollector.cfg,
4. start nwlogcollector

```
root 276701 Feb 14 15:49 NwLogcollector.cfg
root 276701 Feb 14 15:33 NwLogcollector.cfg.1
root 412890 Sep 23 05:47 NwLogcollector.cfg.10
root 412705 Sep 22 22:32 NwLogcollector.cfg.11
root 412855 Sep 16 19:43 NwLogcollector.cfg.12
root 412705 Aug 7 2015 NwLogcollector.cfg.13
root 412705 Jul 7 2015 NwLogcollector.cfg.14
root 412705 Jul 2 2015 NwLogcollector.cfg.15
root 412705 Jul 1 2015 NwLogcollector.cfg.16
root 404031 Jul 1 2015 NwLogcollector.cfg.17
root 404031 Jun 28 2015 NwLogcollector.cfg.18
root 404031 Jun 23 2015 NwLogcollector.cfg.19
root 276701 Feb 14 15:24 NwLogcollector.cfg.2
root 386677 Jun 15 2015 NwLogcollector.cfg.20
root 386677 Jun 15 2015 NwLogcollector.cfg.21
root 371817 Jun 2 2015 NwLogcollector.cfg.22
root 371817 Jun 1 2015 NwLogcollector.cfg.23
root 371817 May 29 2015 NwLogcollector.cfg.24
root 371801 May 26 2015 NwLogcollector.cfg.25
root 430665 Feb 2 21:13 NwLogcollector.cfg.3
root 420476 Dec 12 17:17 NwLogcollector.cfg.4
root 420476 Dec 11 22:43 NwLogcollector.cfg.5
root 420476 Nov 25 19:18 NwLogcollector.cfg.6
root 416138 Oct 9 16:21 NwLogcollector.cfg.7
root 416288 Oct 9 15:33 NwLogcollector.cfg.8
root 412890 Sep 26 15:35 NwLogcollector.cfg.9
```



# File Collection

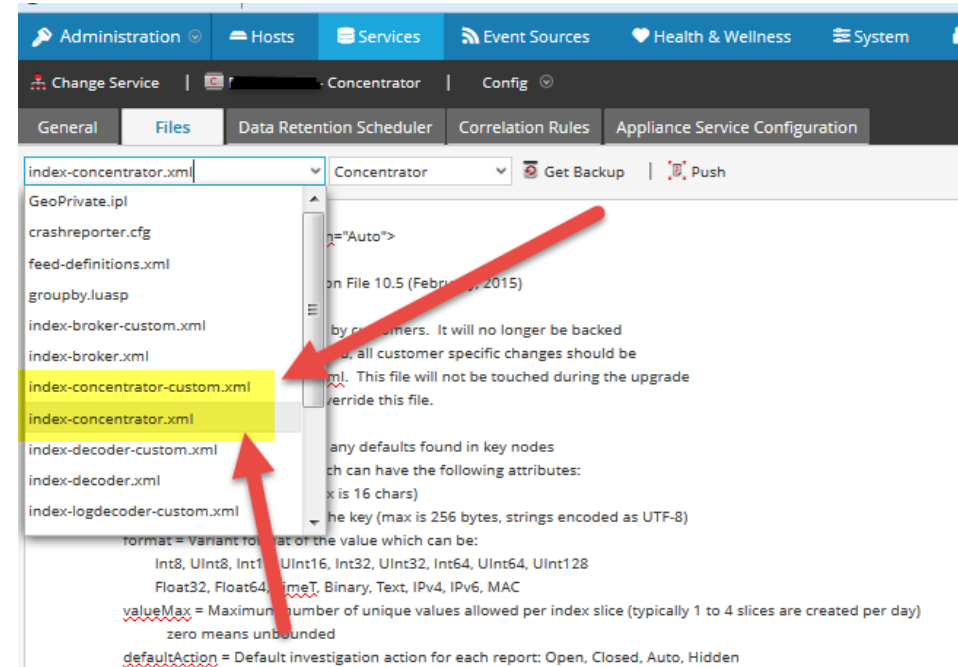
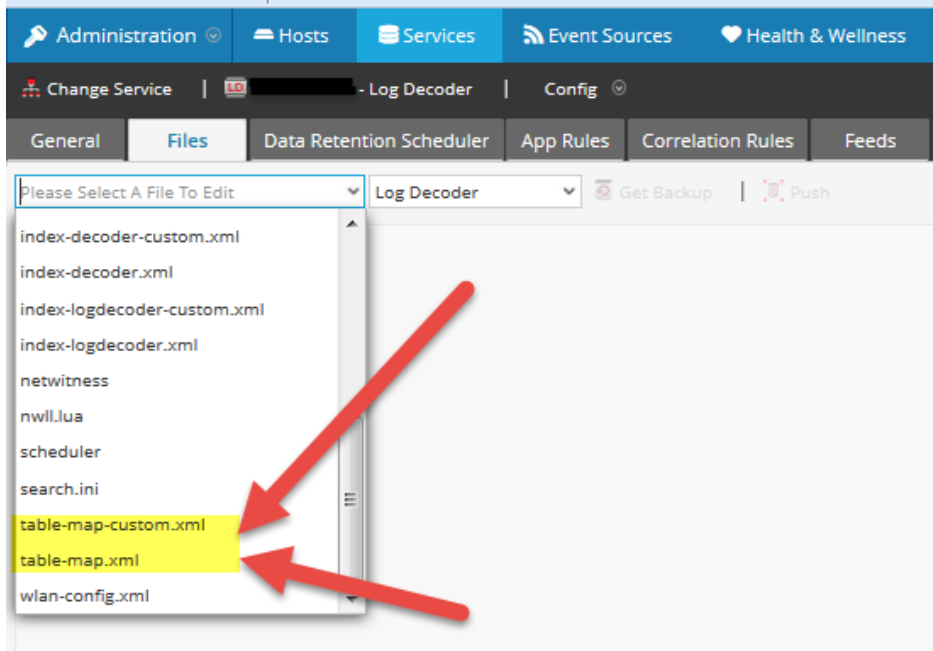
## Need to setup / configure sftpageant

- RSA Agent : [https://sadoes.emc.com/0\\_en-us/300\\_RSA\\_ContentAndResources/02\\_SFTP\\_Agent](https://sadoes.emc.com/0_en-us/300_RSA_ContentAndResources/02_SFTP_Agent)
- Nxlog : <https://nxlog.co/>
- Snare : <https://www.intersectalliance.com/our-product/>

# Table & Concentrator Maps

**Table maps** – primarily used for translating legacy envision tags in parser into NWDB tag names (table-map)

**Concentrator maps** – configuring index values



# Log Decoder Table Maps

- Custom Table Map – /etc/netwitness/ng/envision/etc/table-map-custom.xml
- Table Map (DON'T TOUCH)- /etc/netwitness/ng/envision/etc/table-map.xml
  - **Null Token** – do not populate if X value is present
  - **Flag values**
    - Flags = transient -- not storing this
    - Flags = NONE -- storing this
  - **Data casting**
    - Format = text,ipv4,uint32
  - **envisionDisplayName** –
    - This is used for a performance enhancement in the IPDB extractor connector. Only applicable if you are using IPDB (enVision/Legacy) data source for reporting.

## Example:

```
<mapping envisionName="hostip" nwName="ip.addr" flags="None" format="IPv4"
envisionDisplayName="ComputerIP|IPAddress" failureKey="ipv6.addr" nullTokens="(null)|-"/>
```

**Resource :** [https://sadocs.emc.com/0\\_en-us/089\\_105InfCtr/120\\_AppSerCon/00\\_ApSvGtSt/30\\_AddIServProc/EditCoreServConfFil/UpdtTbIMp](https://sadocs.emc.com/0_en-us/089_105InfCtr/120_AppSerCon/00_ApSvGtSt/30_AddIServProc/EditCoreServConfFil/UpdtTbIMp)



# Log Concentrator Maps

- Custom concentrator Map – /etc/netwitness/ng/envision/etc/index-concentrator-custom.xml
- Concentrator Map (DON'T TOUCH)– /etc/netwitness/ng/index-concentrator.xml
  - **Key description** = A human-readable description for the meta type level
  - **Level** - check link above
  - **valueMax** - defines the maximum number of unique values a key can assume for each time slice.
    - platform has to be told how many unique values each meta key is supposed to hold until the index is saved next (which is every 8 hours by default)
    - When the buffer is full, newest data are prevented to enter the database
  - **the name** - The actual meta key name. Not a pretty name.
  - **the key description** - Represents the "pretty name" value shown in the Investigation screen and other UI components when selecting meta keys.

## Example :

```
<key description="Errors" level="IndexValues" name="error" format="Text" valueMax="50000" />
```

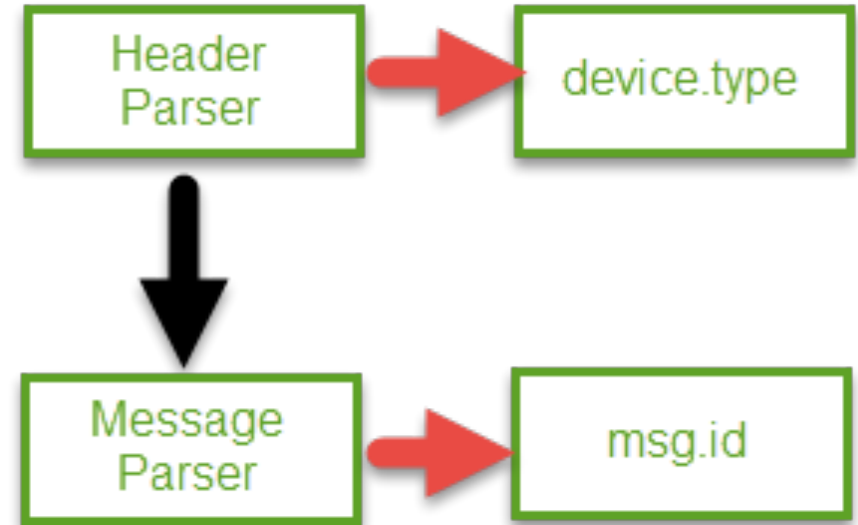
**Resource :** [https://sadoes.emc.com/0\\_en-us/089\\_105InfCtr/120\\_AppSerCon/SACorDbTun/40\\_IndxCust](https://sadoes.emc.com/0_en-us/089_105InfCtr/120_AppSerCon/SACorDbTun/40_IndxCust)

# How To Normalize

- Log Decoder
  - Parsers live here : /etc/netwitness/ng/envision/etc/devices/
  - Each DIR contain required folders INI & XML file. INI contains information about the XML which contains the parsers
    - parserDIRNAME/parsernameMSG.xml
    - parserDIRNAME/parsername.ini
  - New parsers have naming convention prepended:
    - EX : v20\_winevent\_nicmsg.xml
    - ensure device= 2.0 is included for identification by decoder
- Minimum required fields in the INI file for parsers :
  - **devicename.ini**—configuration parameters for the device. You can modify the DisplayName= parameter to a custom value. Do not modify the DatabaseName= parameter.
  - **devicenamemsg.xml**—formatted XML file containing the interpretation code for all of the device messages.
  - **devicenameclient.txt**—text file containing your custom alert description and recommended actions. Edit this file using the enVision UI after adding the new device. For information on editing the alert description and action, see the Manage Messages section in the Help.
  - **devicenamevendor.txt**—text file containing vendor-supplied alert descriptions and recommended actions.

# How To Normalize

- Parsers have two main portions
  - **header parsers**
    - identification of the device (device.type)
    - events are identified top > down (ordering matters)
  - **message parsers**
    - Actual normalization of the event (msg.id)





# How To Normalize

- RSA uses proprietary static text based XML format
  - Temporary Tags -- &lt;fldXX&gt; &lt;hfldXX&gt;
  - Ex: &lt;hfld1&gt;
- The following are the entity reference for all the predefined entities:
  - &lt; <(opening angle bracket)
  - &gt; >(closing angle bracket)
  - &amp; &(ampersand)
  - &quot; "(double quotation mark)
- IDs should technically be the same, but can be different
  - Id1 == message id (SA)
  - Id2 == string matching for message id
- Dave Glover ESI Walkthrough : [https://youtu.be/\\_F1rZc2qLc](https://youtu.be/_F1rZc2qLc)
- Log Parser Guide : <https://community.rsa.com/message/870328>

# How To Normalize

- **HEADER**

- Types of logs
- Event categorization

- **MESSAGE**

- Defines what you want to normalize

- **FUNCTIONS**

- Is where enrichment occurs
- Event time -- normalize different types of events
- URL Function -- take url as input and break into different chunks -- (domain,path,query,scheme,port,etc...)
- Value map function -- kind of like a csv feed, key/value map pairing
- IIS is good example of using/leveraging this. IN IIS, get the raw log and extract the fields you want to chunk out.
- You have to leverage the tagvalmap and set the delimiter, value delimiter
  - Look at voltagesecuredata parser

# How To Normalize – quick modification

- FireEye Log Example (**Request= is not parsing**):

- fenotify-368017.crit: LEEF:1.0|FireEye|MPS|1.0|malware-callback|sev=7^vlan=0^proto=tcp^dstPort=80^cncPort=80^srcPort=6287^externalId=368017^sid=600922^action=blocked^dvchost=sampleBox^src=1.2.3.4^dvc=1.1.2.3^dst=1.3.2.1^cncHost=1.4.3.2^sname=Local.Callback^srcMAC=00:22:33:44:55:66^dstMAC=11:22:33:55:ad:99^devTime=Sep 12 2016 15:32:19 UTC^request=hxxp://www.badSite.com/tmp.exe^link=https://1.1.2.2/event\_stream/events\_for\_bot?ev\_id\=12345&lms\_iden\=00:22:33:44:AA:50^cncChannel=GET http://www.badSite.com/tmp.exe HTTP/1.1::~~Accept: \*/\*::~~Accept-Encoding: gzip, deflate::~~User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; Media Center PC 6.0; .NET4.0C; InfoPath.3)::~~Host: www.badSite.com::~~Proxy-Connection: Keep-Alive::~~::~~^



# How To Normalize – quick modification

1. Identify parser name (**device.type**)

**device.type** = "fireeyewebmps" ⌵

2. Identify header id (**hdr.id**)

**header.id** = "0001"

3. Identify message parser (**msg.id**)

**msg.id** = "malware-callback" ⌵

# How To Normalize – quick modification


```
<VERSION
  xml="13.1"
  checksum="074b88eee71259a5425b073956863eb0"
  revision="94"
  enVision="35010000"
  device="2.0"/>

<TAGVALMAP
  delimiter="^"/>

<HEADER
  id1="0001"
  id2="0001"
  content="&lt;hfld1&gt;-&lt;hfld2&gt;.&lt;hfld3&gt;; LEEF:&lt;hfld4&gt;|FireEye|&lt;hproduct&gt;|&lt;hversion&gt;|&lt;messageid&gt;|&lt;!payload&gt;" />

<HEADER
  id1="0002"
  id2="0002"
  messageId="STRCAT (&apos;Web&apos;, &apos;_&apos;, &apos;MPS&apos;)"
  content="&lt;hfld1&gt;-&lt;hfld2&gt;.&lt;hfld3&gt;; LEEF:&lt;hfld4&gt;|FireEye|Web MPS|&lt;hversion&gt;|&lt;hid&gt;|&lt;!payload&gt;" />

<MESSAGE
```



# How To Normalize – quick modification

```
<MESSAGE
  level="3"
  parse="1"
  parsedefvalue="1"
  tableid="84"
  id1="malware-callback"
  id2="malware-callback"
  eventcategory="1003000000"
  tagval="true"
  content="&lt;@domain:*URL($DOMAIN,url) &gt;&lt;@web_domain:*URL($FQDN,url) &gt;&lt;@event_time:*EVNTTIME($MSG,'%B %F %W %H:%U:%S','%B %F %W
  %H:%U:%O',fld8) &gt;&lt;@msg:*PARMVAL($MSG) &gt;&lt;@severity:*HDR(hfld3) &gt;&lt;@product:*HDR(hproduct) &gt;&lt;@version:*HDR(hversion) &gt;src=&lt;saddr&gt;^
  sname=&lt;signame&gt;^dstMAC=&lt;dmacaddr&gt;^proto=&lt;protocol&gt;^dvchost=&lt;hostname&gt;^vlan=&lt;vlan&gt;^srcPort=&lt;sport&gt;^dvc=&lt;hostip&gt;^cn
  cHost=&lt;dhost&gt;^externalId=&lt;id&gt;^devTime=&lt;fld8&gt;^sid=&lt;sigid&gt;^cncPort=&lt;dtransport&gt;^link=&lt;url&gt;^srcMAC=&lt;smacaddr&gt;^dst=&lt;l
  t;daddr&gt;^dstPort=&lt;dport&gt;^cncChannel=&lt;info&gt;^fileHash=&lt;fld4&gt;^filePath=&lt;filename&gt;^osinfo=&lt;os&gt;^targetApp=&lt;application&gt;^r
  equest=&lt;webpage&gt;^anomaly=&lt;event_description&gt;^user=&lt;c_username&gt;^duser=&lt;username&gt;^msg=&lt;fld7&gt;^shost=&lt;shost&gt;^sev=&lt;fld1&
  gt;^action=&lt;action&gt; " />
```





# How To Normalize – quick modification

```
1 <hfld1>;-<hfld2>;.<hfld3>;: LEEF:<hfld4>;|FireEye|<hproduct>;|<hversion>;|<messageid>;|"<@domain:*URL($DOMAIN,url)>;<@web  
2 fenotify-368017.crit: LEEF:1.0 |FireEye|MPS|1.0 |malware-callback |sev=7^vlan=0^proto=tcp^dstPort=80^cncPort  
3
```

```
.on>;^request=<webpage>;^anomaly=<event_description  
|request=hxxp://www.badSite.com/tmp.exe^link=https://1
```

# How To Normalize – quick modification

- If parser is ok, check the following items :
  - Log decoder Table Map (RSA supported table)
    - **services >> files >> table-map.xml**
  - CUSTOM log decoder map to update if rsa does not support (CUSTOM Supported table)
    - **services >> files >> table-map-custom.xml**

# How To Normalize – quick modification

- Steps to troubleshoot
  1. Flip the tag in the parser to enable (if temporary tag) / FIX XML parser
  2. if parser supports, check the table map on decoder to see if tag is enabled
    - Flags='Transient' == not enabled
    - Flags='None' == Enabled

```
<mapping envisionName="web_root" nwName="web.root" flags="Transient" envisionDisplayName="WebRoot|RootDomain"/>  
<mapping envisionName="webpage" nwName="web.page" flags="Transient" envisionDisplayName="WebPage"/>  
<mapping envisionName="wifi_channel" nwName="wlan.channel" flags="Transient" format="UInt16"/>  
<mapping envisionName="wlan" nwName="wlan.name" flags="Transient" envisionDisplayName="WLAN"/>
```

NOTE : After updating table map / Parser , restart the log decoder service (restart nwlogdecoder )

```
<mapping envisionName="web_domain" nwName="web.domain" flags="None" envisionDisplayName="WebDomain"/>  
<mapping envisionName="webpage" nwName="web.page" flags="None" envisionDisplayName="WebPage"/>  
<mapping envisionName="web_query" nwName="query" flags="None" envisionDisplayName="WebQuery"/>
```

# Common Event Format

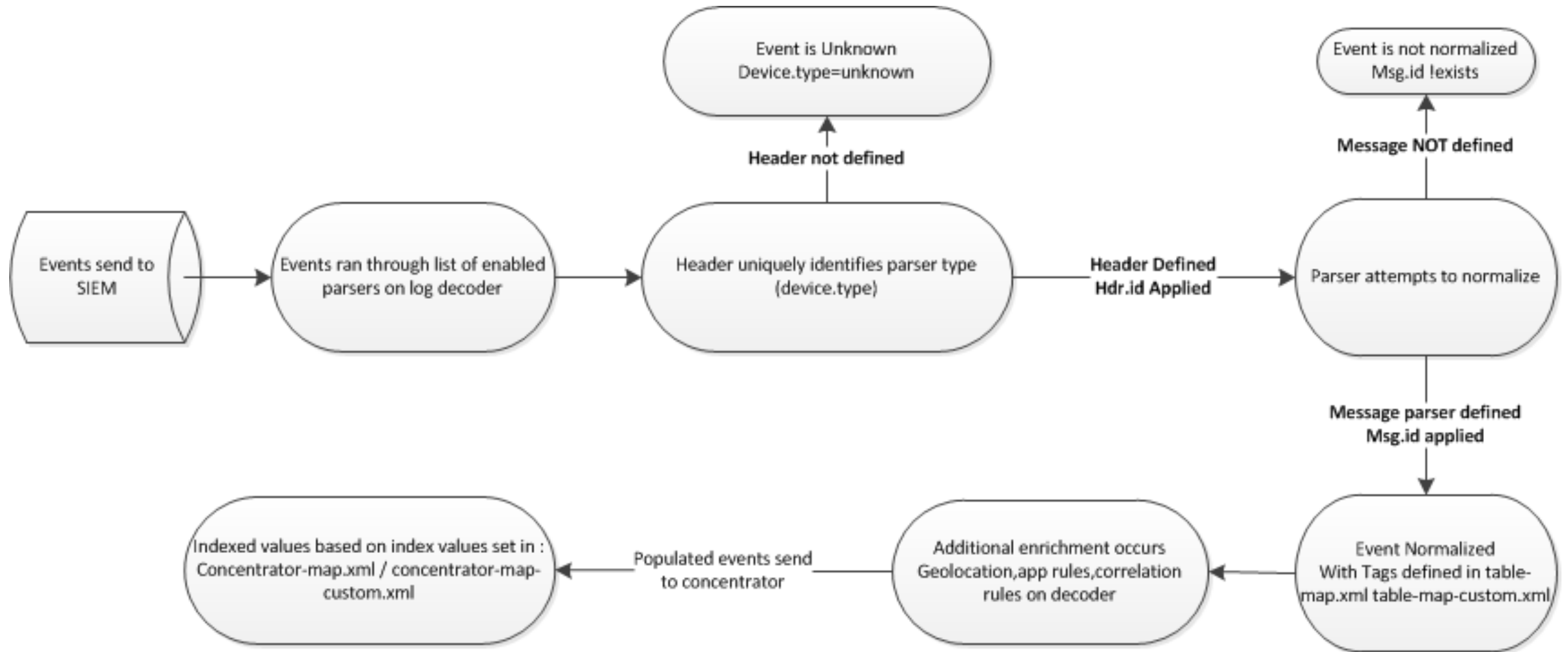
- Common event format
  - Events pre-normalized into SIEM.
- CEF Parser named cef in CEF folder under */etc/netwitness/ng/envision/etc/devices/cef*
- CEF events device.type name are dynamic (what the app generates)

```
<ExtensionKeys>
  <ExtensionKey cefName="act" metaName="action"/>
  <ExtensionKey cefName="app" metaName="application"/>
  <ExtensionKey cefName="cat" metaName="category"/>
  <ExtensionKey cefName="cnt" metaName="event_counter"/>
  <ExtensionKey cefName="destinationDnsDomain" metaName="ddomain"/>
  <ExtensionKey cefName="destinationServiceName" metaName="service"/>
  <ExtensionKey cefName="destinationTranslatedAddress" metaName="dtransaddr"/>
```

```
<ExtensionKey cefName="cs5" metaName="cs_fld">
  <device2meta device="mcafeeewg" metaName="policyname" label="Policy"/>
  <device2meta device="bit9" metaName="rulename" label="ruleName"/>
</ExtensionKey>
```

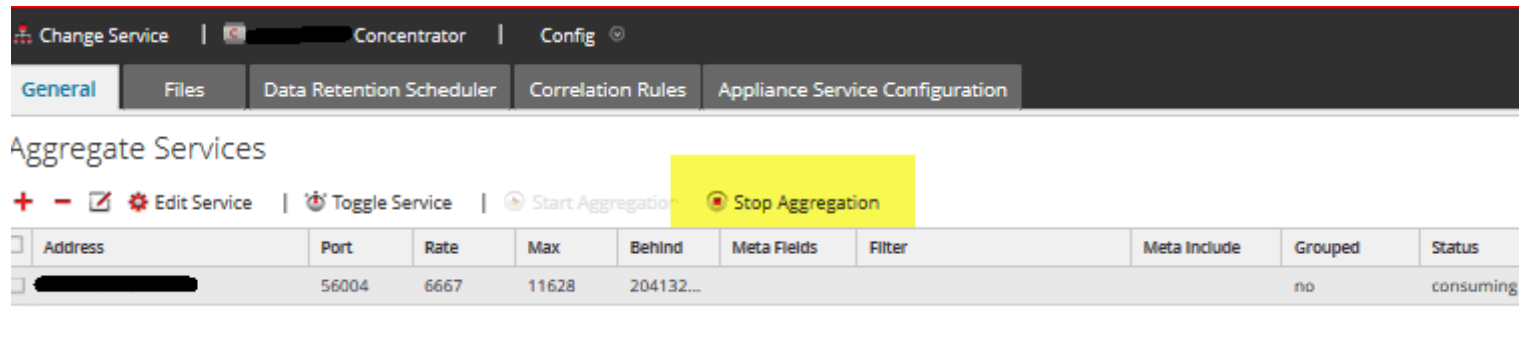


# Parser Content - Workflow



# Graceful concentrator reloading

- Steps to properly restart the concentrator 'gracefully'
  - If not, concentrator may force a reindex, which could take several hours.
- 1) Stop aggregation on the concentrator



2) Tail the logs on the concentrator and wait for "Index Save Completed "

```
Aug 9 18:51:23 testConcentrator NwConcentrator[5608]: [Index] [info] Index save completed
```

3) restart the concentrator service. >> restart nwconcentrator

# Testing / Validating Parser

- ESI Beta Tool

- XMLLINT

This is to help guide users on how to validate whether their xml is valid syntax. This may not be specific RSA XML syntax, just general xml syntax in general. If the output shows empty, then the syntax is correct. Else this will populate errors

– **Reference** : [https://en.wikipedia.org/wiki/XML\\_validation](https://en.wikipedia.org/wiki/XML_validation)

```
t;saddr> &lt;port>| &lt;saddr>}" />  
  
</DEVICEMESSAGES>  
[root@ _ _ _ _ ciscoportwsa]# xmllint v20_ciscoportwsamsg.xml -noout
```

- Notepad ++

# Testing / Validating Parser

- **replay/inject events**

- nwlogplayer.exe (default package on SA appliances) > sends events via syslog

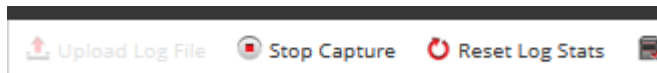
- Syntax :

- Nwlogplayer.exe -s IPADDR -r X -f <LOGFILENAME>

- **Reference:** <https://community.rsa.com/docs/DOC-43179>

- **GUI Based upload**

- Stop capture, and then upload



**Log Decoder Service Information**

# Parser Deployment

- **new parsers (identified by log directory creation), will be enabled by default**
  - need to restart nologdecoder service for new parsers to be identified, show in GUI
- **existing parsers will keep existing configurations (enabled/disabled)**
  
- **Custom Parser Permissions**
- **Needed to ensure the log decoder properly identifies the parser**
  - `chmod 755 -R /etc/netwitness/ng/envision/etc/devices/*`
  - `chown root:root -R /etc/netwitness/ng/envision/etc/devices/*`



# Log Parser Due Diligence

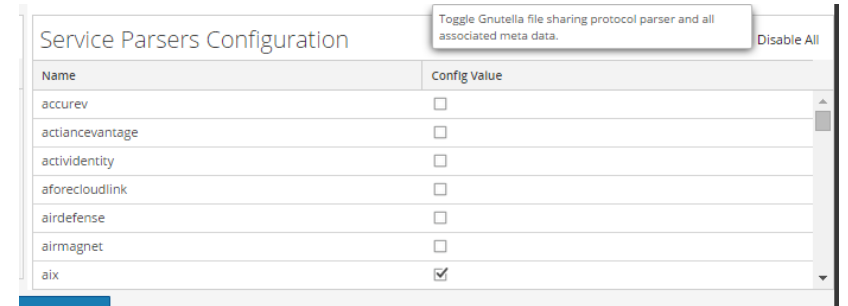
1. Inquire if RSA released content fixes any recent requests submitted on our behalf. (Will help identify fixes/testing needs).
2. Diff check parsers with diff tool / RSA provided script -- identify specific items that have changed from previous versions.
3. Check to ensure pre-existing modifications have been supported by RSA and modify for continued modification, if needed (Tag Changes).
4. If any applicable existing modifications need to be merged, research/merge/carry over modifications into custom parser.
5. If RSA has overtaken existing customized modifications, migrate custom parser in REPO into retired DIR, and push rsa supported parser from LIVE.
6. Push to UAT --
7. inject & validate events to UAT for unknown/misidentification & test content generation(if tag changes occur) issues that should be resolved (if RSA claims they have, see item #1 )
8. ensure UAT log decoder services restart with no errors
9. Fill out / document details in the ISS onenote log parsers changes notebook Fill out change control >> package up **custom** content (XML files) and place on SA head
10. RSA supported content can be pushed directly from LIVE.

# Log Parser Deployment Checklist

1. backup **each** piece of content going to be updated/modified in change control.
2. **Before deploying custom content, ENSURE custom content has proper permissions.**
3. Ensure custom fileTypeSpec/custom parsers have proper permissions
  1. ***chmod 755 -R DIRECTORY***
  2. ***chown root:root -R DIRECTORY***
4. deploy custom parser **FIRST** to any DR appliance
5. if existing parser, need to only deploy XML files.(ini should already be deployed)
6. else see next steps below : (New Custom Parser Deployment Details )
7. restart log decoder /appliance service -- **restart nwlogdecoder && tail -f /var/log/messages**
8. ensure it comes up back with no errors.
9. If decoder reloads properly, deploy to all log decoders, and restart each instance
10. *Tail -f /var/log/messages | grep -I 'warning|failure|error|successfully'*
11. *Tail -f /var/log/messages | grep -I 'logparse'*
12. inject events if needed to validate parser working as intended, **if needed** (Specific use cases/needs)
13. Do a quick query against events for each device.type that are identified>> via the webui, **if needed**.
14. Check concentrators to identify if decoders are online, and consuming
15. backup content into appropriate GIT directories.

# New Custom Parser Deployment

- **This is for custom/modified parsers that do not currently exist within the log decoder.**
- Create directory on the log decoder in the appropriate log directory (**start with DR decoder to test**)
  - Make directory on log decoder -- `mkdir /etc/netwitness/ng/envision/etc/devices/DEVICE_NAME`
- Push out the appropriate .ini and .xml parser files into this newly created parser directory
- Since these are new log directories, the log decoder service will need to be restarted in order for it to properly identify the parser.
  - **NOTE:** new parsers will automatically be enabled by default. Need to disable parsers on those decoders that do not need to be enabled
- Stop Service -- `stop nwlogdecoder`
- Start log decoder service -- `start nwlogdecoder`
- Monitor /var/log/messages for any warning/failures/errors.
  - `Tail -f /var/log/messages | grep -I 'warning|failure|error|successfully'`
  - You will note any errors, and should identify when the following message populates:
    - **Module logdecoder successfully loaded**
    - **You should see specific parsers : loaded successfully message in /var/log/messages**
- You will note within the messages file that the log parser properly identified by noting PARSENAME parser loaded within the log file.
  - You can also check by going into the Log Decoder GUI config page and identify whether the parser has loaded:
    - **NOTE:** There are issues with current versions of SA, that do not show disabled parsers consistently. You will have to go into the disabled.list within the explorer page on the log decoders. This will help if you deploy the parser, and it doesn't show up in the log decoder, and disabled.
- When the log decoder service restarts, it'll identify any / all new identified directories with parser inside as enabled. You will want to disable them in the previous picture, and reload the parsers via a variety of ways shown here :



Name	Config Value
accurev	<input type="checkbox"/>
actiancevantage	<input type="checkbox"/>
actidentity	<input type="checkbox"/>
aforecloudlink	<input type="checkbox"/>
airdefense	<input type="checkbox"/>
airmagnet	<input type="checkbox"/>
aix	<input checked="" type="checkbox"/>

# Existing Parser Deployment

- **This is for parsers that already exist on the decoder, just need an update/upgrade.**
- Copy/Push parser to appropriate existing directory on the log decoder
- Reload the parser instance in a variety of ways shown above
- Monitor `/var/log/messages` for any warning/failures/errors.
  - `Tail -f /var/log/messages | grep -I 'warning\\|failure\\|error\\|successfully'`
  - You will note any errors, and should identify when the following message populates:
    - **Module logdecoder successfully loaded**

# Parser Due Diligence

- ALWAYS check parsers when updating
  - Use the following to identify differences/changes in versions:
    - **kdiff**
    - **winmerge**
    - **linux diff**
    - **content engineering script demo** : new ESI format is not supported format only
- Keep/maintain local repo of parsers like GIT to reference previous parser versions
- Lookout for the content file type that RSA releases (bulk parser release)



# RSA Live parser packages

envision == ZIP

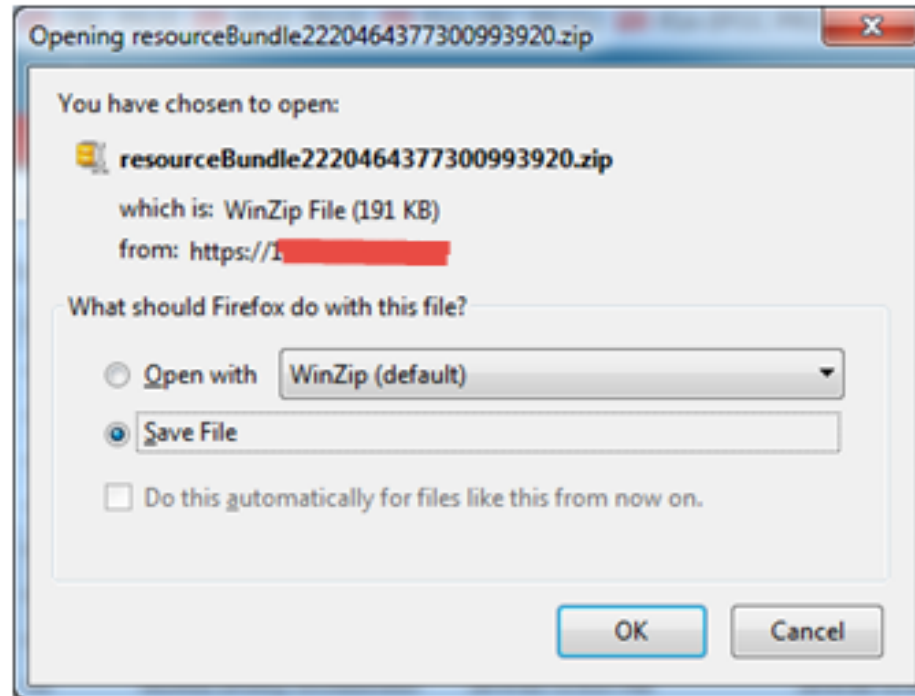
File compression tools can uncompress  
Adjust file extension .envision with  
7zip, winzip and unzip.

The screenshot shows the 'Search Criteria' sidebar in the RSA Live interface. It includes fields for 'Keywords', 'Resource Types' (with 'RSA Log Device' selected), 'Medium', 'Tags', 'Required Meta Keys', and 'Generated Meta Values'. At the bottom, there are date pickers for 'Resource Created Date' and 'Resource Modified Date', and 'Search' and 'Cancel' buttons.

The screenshot shows the 'Matching Resources' table with a context menu open over the 'Cisco ASA' row. The context menu has 'Create' and 'Deploy' options. Red arrows point from the text above to the 'envision == ZIP' text and the 'Create' menu item.

<input type="checkbox"/>	Subscriber	Name	Created		Type	Description
<input type="checkbox"/>	no	Cisco Secure ACS Appliance	2014-02-13 9:26 PM	2016-08-17 9:49 AM	RSA Log Device	Log device content for ev
<input type="checkbox"/>	no	Windows Events (NIC)	2014-02-13 9:55 PM	2016-08-17 3:12 PM	RSA Log Device	Log device content for ev
<input type="checkbox"/>	no	Fortinet Manager/Analyzer	2014-02-13 9:32 PM	2016-08-17 9:09 AM	RSA Log Device	Log device content for ev
<input type="checkbox"/>	no	Fortinet FortiGate	2014-02-13 9:32 PM	2016-08-17 9:09 AM	RSA Log Device	Log device content for ev
<input checked="" type="checkbox"/>	no	Cisco ASA	2014-02-13 9:24 PM	2016-08-17 9:09 AM	RSA Log Device	Log device content for c
<input checked="" type="checkbox"/>	no	Windows Events (Snare)	2014-02-13 9:55 PM	2016-08-16 8:01 AM	RSA Log Device	Log device content for c
<input checked="" type="checkbox"/>	no	Microsoft Internet Authentic...	2015-10-18 12:14 PM	2016-08-16 8:01 AM	RSA Log Device	Log device content for c
<input checked="" type="checkbox"/>	no	McAfee ePolicy Orchestrator	2014-02-13 9:31 PM	2016-08-16 8:01 AM	RSA Log Device	Log device content for c
<input type="checkbox"/>	no	Barracuda Spam Firewall	2014-02-13 9:22 PM	2016-08-16 8:01 AM	RSA Log Device	Log device content for ev
<input type="checkbox"/>	no	FireEye WebMPS	2014-02-13 9:31 PM	2016-08-15 4:38 PM	RSA Log Device	Log device content for ev
<input type="checkbox"/>	no	Iuniner SSI VPN	2014-02-13 9:37 PM	2016-08-14 9:55 AM	RSA Log Device	Log device content for ev

# RSA Live parser packages



# RSA Live parser packages

The screenshot displays the RSA Live interface for deploying a resource package. The main window is titled "Resource Package Deployment" and shows a progress bar for the deployment of "Windows Events (NIC)". The deployment is successful, as indicated by the message "Live deployment task finished successfully".

**Log Decoder Instance reloads automatically after uploads**

Service Name	Resource Name	Status	Progress
Lo...	Windows Events (NIC)	5 of 5	<div style="width: 100%; height: 10px; background-color: green;"></div>

The interface also shows a list of "Matching Resources" on the left and a "Deploy" button at the bottom right, which is highlighted by a red arrow.

# Parser Monitoring

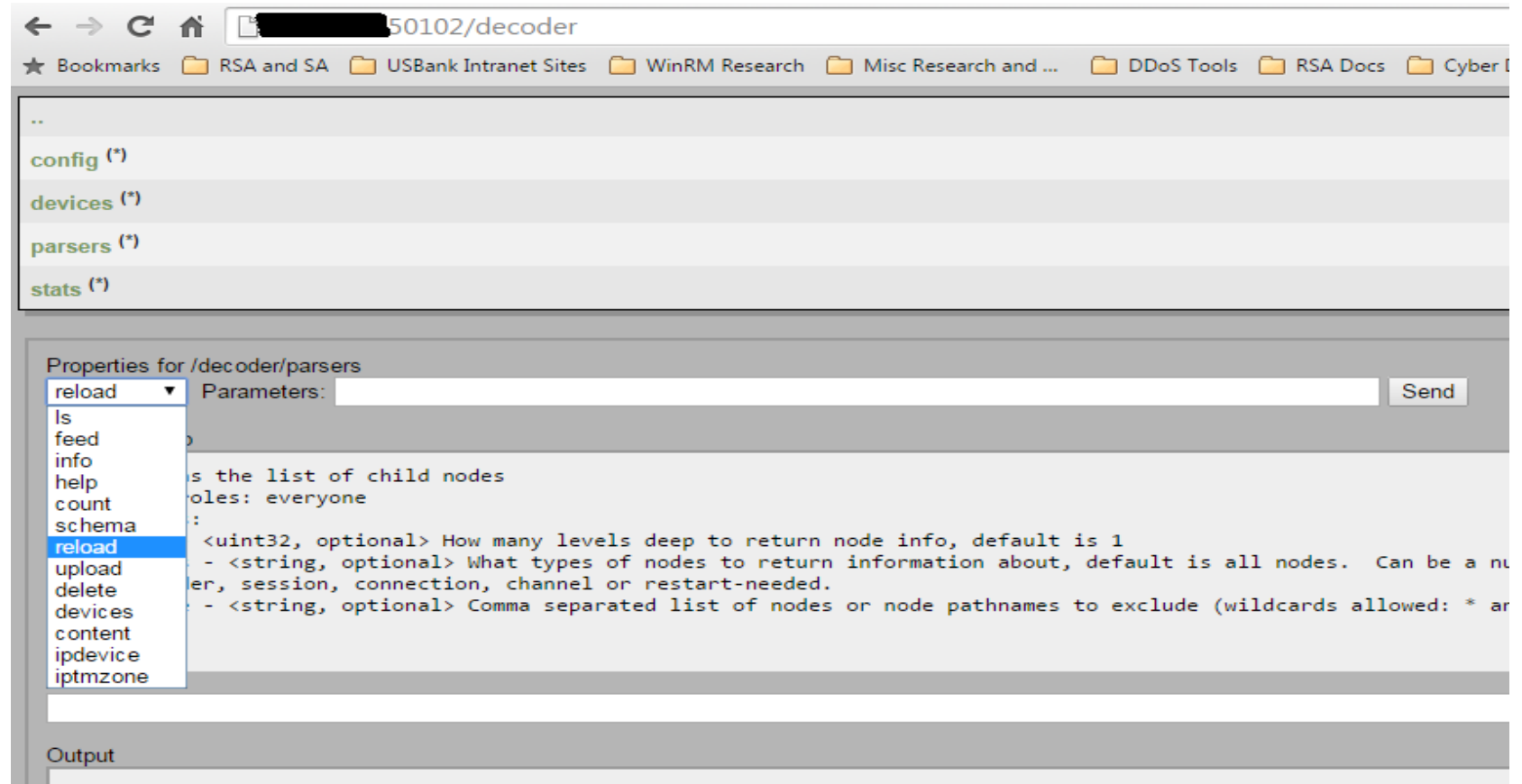
- **msg="has requested the schema for loaded parsers"**
- Jun 09 2016 22:57:59 testDecoder CEF:0|RSA|Security Analytics Audit|10.5.2.0|MANAGEMENT|schema|6|rt=Jun 09 2016 22:57:59 src=1.1.1.1 spt=49184 **suser=joegumke** sourceServiceName=LOG\_DECODER deviceExternalId=f959d2b0-6de3-4ec9-aa71-8c8229275d2d deviceProcessName=NwLogDecoder outcome=pending **msg=has requested the schema for loaded parsers**
- You can partially monitor for parser changes with Global Auditing in SIEM
  - Key Events :
    - User has uploaded files X,Y,Z through rsa live
    - User has issues a parser reload.

# Reloading parsers

1. REST GUI
2. REST API
3. Log Decoder Explorer
4. Log Decoder Enable/Disable GUI
5. Command Line
6. Nwconsole Command line
7. Uploading RSA Live package

# Reloading parsers Part 1

- **REST Call GUI** : [https://log\\_decoder::50102](https://log_decoder::50102)
- Log Decoder > Parser (Click on \* next to parser):
- **Reload >> Send**





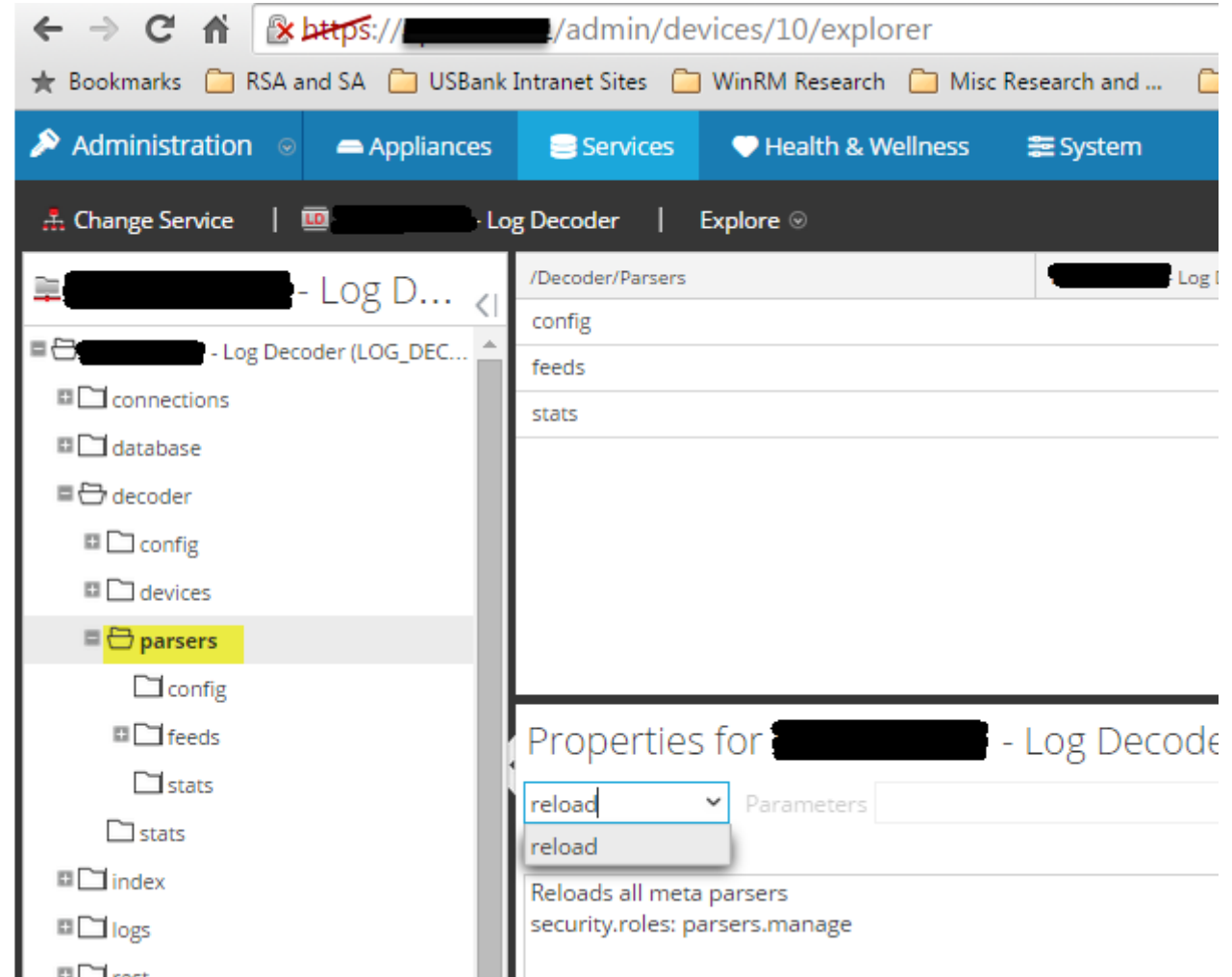
# Reloading parsers Part 2

- **Rest Call REST API –**
- `LogdecoderIP::50102/decoder/parsers?msg=reload&force-content-type=text/plain&expiry=600`

# Reloading parsers part 3

- **Log Decoder Explorer :**

Find the following path in the explorer path, and right click on PARSERS, and go to properties.



# Reloading parsers part 4

The screenshot displays the RSA Charge 2016 configuration interface for a Log Decoder service. The interface is divided into several sections:

- System Configuration:** A table with columns 'Name' and 'Config Value'.

Name	Config Value
Compression	0
Port	50002
SSL FIPS Mode	<input type="checkbox"/>
SSL Port	56002
Stat Update Interval	1000
Threads	15
- Log Decoder Configuration:** A table with columns 'Name' and 'Config Value'.

Name	Config Value
<b>Adapter</b>	
Berkley Packet Filter	
Capture Interface Selected	log_events,Log Events
<b>Cache</b>	
Cache Directory	/var/netwitness/logdecoder/cache
Cache Size	4 GB
<b>Capture Settings</b>	
- Parsers Configuration:** A table with columns 'Name' and 'Config Value'.

Name	Config Value
<input checked="" type="checkbox"/> ALERTS	<input checked="" type="checkbox"/>
BITTORRENT	<input type="checkbox"/>
<input checked="" type="checkbox"/> FeedParser	<input checked="" type="checkbox"/>
	<input type="checkbox"/>
	<input checked="" type="checkbox"/>
GNUTELLA	<input type="checkbox"/>
- Service Parsers Configuration:** A table with columns 'Name' and 'Config value'.

Name	Config value
postgresql	<input type="checkbox"/>
proofpoint	<input type="checkbox"/>
radwaredp	<input type="checkbox"/>
resetme	<input checked="" type="checkbox"/>
rhlinux	<input checked="" type="checkbox"/>
riverbedsteelhead	<input type="checkbox"/>
rsaaah	<input type="checkbox"/>

A tooltip is visible over the 'SSL FIPS Mode' checkbox, stating: "Determines whether the OpenSSL library will enter FIPS mode. Change takes effect on service restart."

An "Apply" button is located at the bottom center of the configuration area.

# Reloading parsers part 5

- Command Line :
  - Go into the log decoder and run :
    - Start nwlogdecoder
    - Stop nwlogdeocder
    - Restart nwlogdecoder

# Reloading parsers part 6

- NWConsole Command Line :
  - NwConsole -c login localhost:56002:ssl <user> <password> -c decoder/parsers reload

# Reloading parsers part 6

- RSA live package
  - More efficient method to deploying rsa/custom content.
  - Steps :
    - 1. Deploy custom content
    - 2. Deploy rsa package
      - Parser instances will reload, and have to check each decoder for failure/warning
        - RELOADING parser instance, DOES NOT show successful reload like restart nwlogdecoder service



# How To Troubleshoot BOOST Failure

```
Oct 6 16:56:56 test-logDecoder nw[3269]: [LogParse] [warning] Message parse failure in file ciscoiportwsa: Throw location unknown (consider using BOOST_THROW_EXCEPTION)Dynamic exception type: boost::exception detail::clone impl<boost::exception detail::error_info injector<boost::spirit::qi::expectation_failure<_gnu_cxx::__normal_iterator<char const*, std::string> > > >std::exception::what: boost::spirit::qi::expectation_failure[nw::envision::(anonymous namespace)::content tag*] = <@domain:*URL($DOMAIN,url)><@web domain:*URL($DOMAIN,url)><@web root:*URL($ROOT,url)><@webpage:*URL($PAGE,url)><@:*SYSVAL($MSGID,$ID1)><@event time:*EVNTTIME($MSG,'%D/%B/%W:%N:%U:%O',fld20)><@msg:*PARMVAL($MSG)><saddr> { { "<fld1>\<c_username>@<fld2>" | <c_username> } @<fld1> | - } <fld2> [<fld20> <timezone>] "<web_method> <url> <version>" <resultcode> <rbytes> <action>:<fld4> <duration_string> <polycname> <<<fld17>,{ ns | <reputation_num> },<fld18>> <fld19>[nw::envision::(anonymous namespace)::expectation_failure_start tag*] = { { "<fld1>\<c_username>@<fld2>" | <c_username> } @<fld1> | - } <fld2> [<fld20> <timezone>] "<web_method> <url> <version>" <resultcode> <rbytes> <action>:<fld4> <duration_string> <polycname> <<<fld17>,{ ns | <reputation_num> },<fld18>> <fld19>[nw::envision::id1_tag*] = CONNECT[nw::envision::id2_tag*] = CONNECT
```

The **blue** tells you what parser the failure is in.

The **gray** gives you a brief boost reason for the failure.

The **pink** tells you the failure occurred in the MESSAGE element content attribute and continues to give you the value for the content attribute.

The **orange** tells you where the failure started in the content attribute. From here, I was able to quickly see that the syntax did not look right.

The **yellow** tells you what MESSAGE element id1 the failure occurred in.

# Log Parser Warning Error

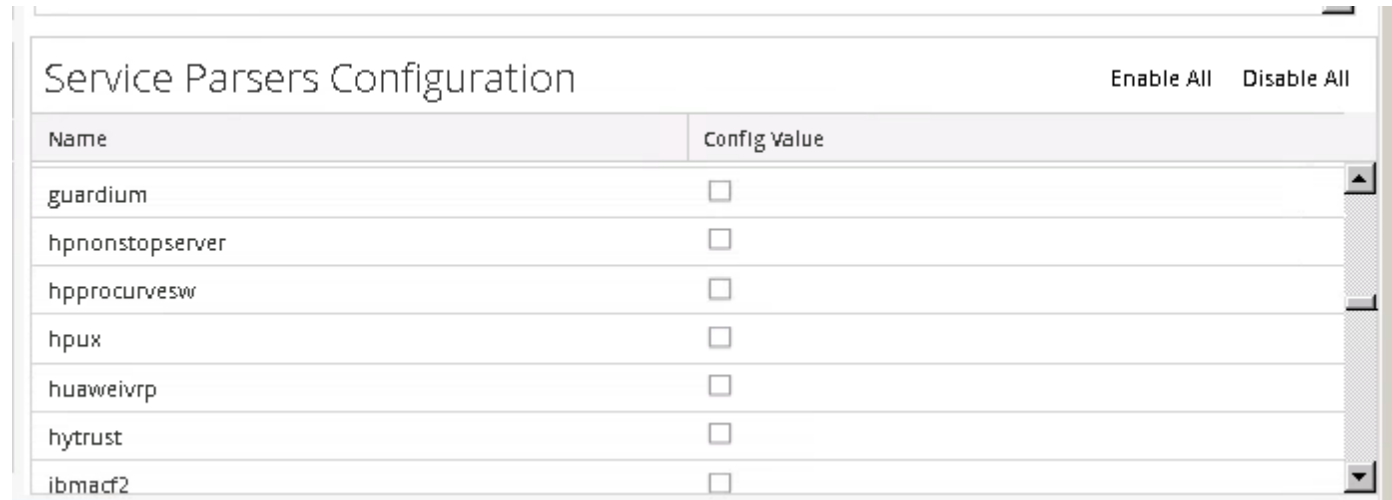
- If you happen to have a typo within the new parser, and restart, you will see the decoder service continually restart over and over. You will notice an initialization error in the GUI on the log decoder system page.
- Note the following error in the `/var/log/messages` on the specific log decoder :
  - Jun 18 20:11:03 epoc-dlt01 nw[9948]: [Engine] [warning] Module logdecoder failed to load: Invalid entity reference **>>**
- See the above bolded characters as they are the culprits regarding the incorrectly typed space in the parser that you will need to go back and fix to resolve this restarting/initialization issue.

# Parser Management

- Managed by parser version number
  - RSA == xml=XXX
  - Custom == xml=XXX.YYY
- scrap xml version numbers to see what version
  - Can use a .YYY notation for custom purposes.
    - Ex : xml="32.001"
  - **Example Query :**
  - for i in \$(cat /tmp/applianceList); do
  - ssh root@\$i 'find /etc/netwitness/ng/envision/etc/devices/ -type f -regex '.\*xml' | xargs grep xml=\'' > \$fileLocation/\$i.txt
  - sed -i 's/\etc\netwitness\ng\envision\etc\devices\\/' \$fileLocation/\$i.txt
  - sed -i 's/[[:blank:]]//g' \$fileLocation/\$i.txt
  - sed -i 's/\/,/' \$fileLocation/\$i.txt
  - sed -i 's/\.xml:\.xml,/' \$fileLocation/\$i.txt
  - done

# Decoder Tips

- DISABLE and/or Remove Unneeded parsers from the log decoder directory (altogether) from the log decoder. Reduce overhead / management of parsers.
  - File Location : /etc/netwitness/ng/envision/etc/devices/PARSERDIR/



The screenshot shows a window titled "Service Parsers Configuration" with two buttons: "Enable All" and "Disable All". Below the title bar is a table with two columns: "Name" and "Config Value". The table lists several parsers, each with an unchecked checkbox in the "Config Value" column.

Name	Config Value
guardium	<input type="checkbox"/>
hpnostopserver	<input type="checkbox"/>
hprocurvesw	<input type="checkbox"/>
hpux	<input type="checkbox"/>
huaweivrp	<input type="checkbox"/>
hytrust	<input type="checkbox"/>
ibmacf2	<input type="checkbox"/>

# Tips / Tricks

- **Unidentified events**
  - (device.type=unknown)
  - device.ip = XXX.XXX.XXX.XXX and device.type=unknown
- **Unidentified message parser**
  - device.type=XXXXXXXX && msg.id !exists
- **Mis-Categorized Events**
  - device.type=XXXXX , AND is supposed to be something else.
- **Parse Error Events**
  - parse.error exists

# Parser Validation Script demos

- Demos
  - Powershell script to validate versions
  - Sh script to extract the xml files
  - RSA diff Script

Please Complete Session Evaluation



The background features a complex digital aesthetic. On the left, there are white, glowing circuit traces that form a grid-like pattern. The right side is dominated by a vertical column of binary code (0s and 1s) in a light blue color. The overall color palette is dark blue to black, with highlights of white and light blue.

# Appendix

## Miscellaneous Parser Information

# Appendix : Nwconsole local testing

- 1. You can run this locally without connecting to the LogDecoder
- 2. Quick command line option to test parsing
- 3. The output you see here is exactly the same as what the LogDecoder produces, unlike the inconsistency issues with ESI. (Make sure your LD and nwconsole.exe versions are the same).
- 4. You can use device=devicetype to test parsing against a particular parser only.
  
- I think this can serve as an interim tool that you always wanted till you get the ESI to test parsers.
- Updated Steps for local execution:-
  - 1. Copy the /etc/netwitness/ng/envision folder from your LogDecoder box. That contains all the required files for parsing.
  - 2. Paste this at the following location on the system where you have nwconsole installed. "C:\ProgramData\NetWitness"
  - 3. Execute nwconsole.exe and wait till you see a prompt.
  - 4. Execute logparse in=C:/test.log metaonly (There are several options available,
    - you can use "out", "device" etc)
  
- Here is an example:-
  - > logparse in=C:/test.log metaonly

# Appendix : Nwconsole commands

- The latest NwConsole logParse command options:
- 
- Usage: logParse in=infile [out=outfile] [content=c2|c3] [device=device-enabled] [path=configpath][metaonly]
- in=infile The input source file. "in=stdin" means user type the log in. Required parameter.
- out=outfile The output file. If not specified, use stdout as output.
- content=c2|c3 Content version, either c2 or c3. Default is c2
- device=device-enabled
- Comma delimited device list specifying devices that is enabled. Default enable all devices
- path=log-parsers-config-path The logparsers configuration path. Default will find configuration file like logdecoder
- metaonly If presented the output will only contains parsed meta, otherwise will print log message after metas
- 
- Bring up the console by command NwConsole, and just type logParse will leads to this help message. However, when you actually use it, you have to invoke it by type "NwConsole -c logParse logParse-options...". The path is the envision directory path. For example, if your envision directory is under /home/username/test, then the value you put for path should be path=/home/username/test/envision.

# Parser Management

Find Versions:

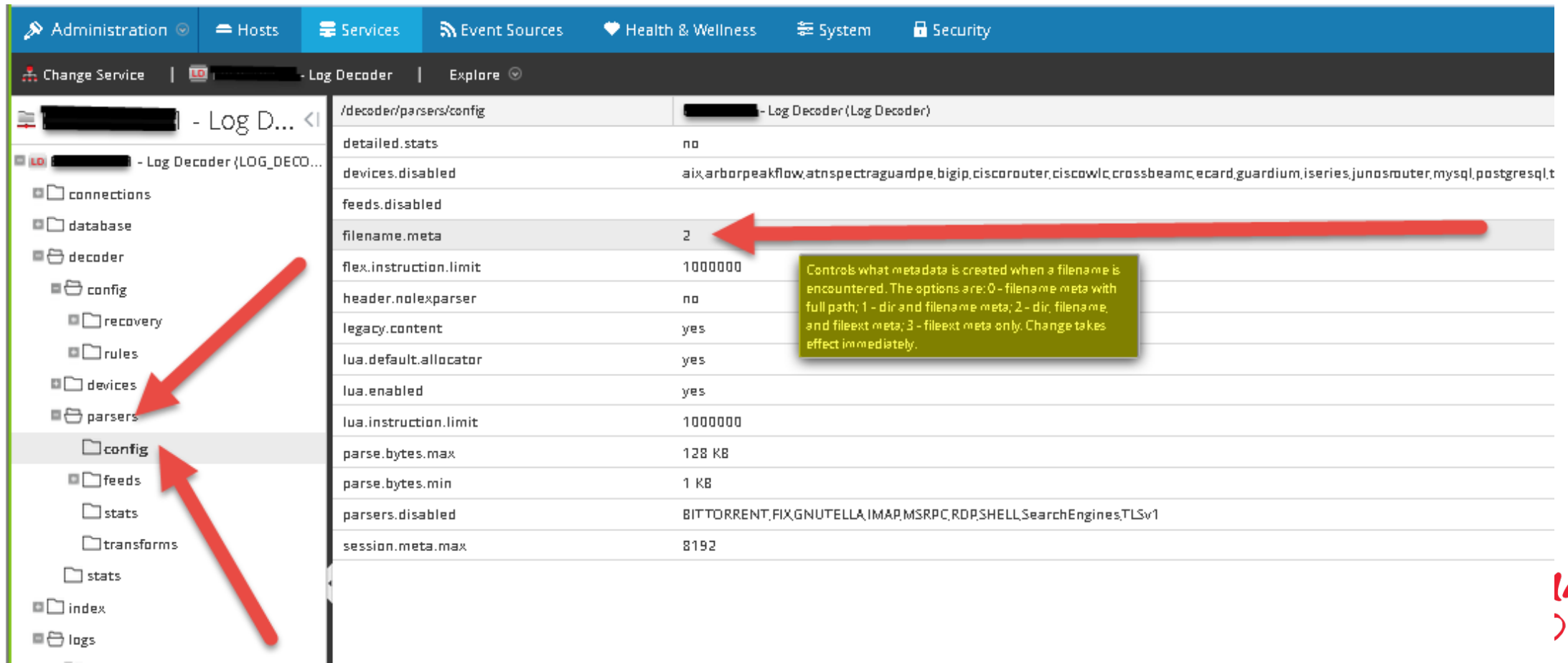
```
ssh root@XXX-dl0X 'find /etc/netwitness/ng/envision/etc/devices/ -type f -regex '.*xml' | xargs grep xml=' > /tmp/XXX-dl0XVersions.txt
```

Find Parsers :

```
ssh root@XXX-dl0X ls /etc/netwitness/ng/envision/etc/devices/ > /tmp/XXX-dl0XDevices.txt
```

# Magic Voodoo Filename Translation

- A parser was strange in the fact that the filename was mysteriously putting frame into the directory and extension keys, even though there was no tablemap translations for this. There is a hidden translation on the log decoder here that is a special configuration setting that takes anything in filename and places them per the setting configured below.



The screenshot shows the Splunk configuration interface for the Log Decoder service. The left sidebar shows a tree view of configuration folders, with 'parsers' and 'config' highlighted. The main panel displays a list of configuration settings for '/decoder/parsers/config'. The 'filename.meta' setting is highlighted with a red arrow and a callout box.

Setting	Value
detailed.stats	no
devices.disabled	aix,arborpeakflow,atnspectraguardpe,bigip,ciscorouter,ciscowlc,crossbeamc,ecard,guardium,iseries,junosmuter,mysql,postgresql
feeds.disabled	
filename.meta	2
flex.instruction.limit	1000000
header.nolexparser	no
legacy.content	yes
lua.default allocator	yes
lua.enabled	yes
lua.instruction.limit	1000000
parse.bytes.max	128 KB
parse.bytes.min	1 KB
parsers.disabled	BITTORRENT,FIX,GNUTELLA,IMAP,MSRPC,RDP,SHELL,SearchEngines,TLSv1
session.meta.max	8192

Controls what metadata is created when a filename is encountered. The options are: 0 - filename meta with full path; 1 - dir and filename meta; 2 - dir, filename, and fileext meta; 3 - fileext meta only. Change takes effect immediately.

large  
216

# Appendix : Tag Value maps

- Saw that the new fireeye parser includes in the infection-match : `missField="true"`. What is this?
- `missField` attribute allows the parser to allow missing tags. So if an incoming log has some tags that are not in the parser, that will be allowed. Basically the parser wouldn't choke and make the entire log unknown if a tag is missing.

# Appendix : MISC Rest commands

## Log Decoder Parser Count :

`http://<logdecoderip>:50102/decoder/parsers/stats/parser.count?msg=get&force-content-type=text/plain&expiry=600`

## Disabled Parsers:

`http://<logdecoderip>:50102/decoder/parsers/config/devices.disabled?msg=get&force-content-type=text/plain&expiry=600`

**This command will wipe out everything on the decoder, so...use it wisely.**

`NwConsole -c login localhost:50004 admin netwitness -c /decoder reset data=1 force=1 -c logout; tail -f /var/log/messages`



# Appendix : LUA Parsers

## Create Parser

```
Local luatest = nw.createParser("lua_test", "TEST PARSER")
```

```
  Create Custom meta key
```

```
Concentrator: index-concentrator-custom.xml
```

```
  <key description="Test Alert" level="IndexValues" name="test.alert" valueMax="1000" format="Text"/>
```

```
  Define Meta Keys to write meta into
```

```
-- Step 2 - Define meta keys to write meta into
```

```
-- declare the meta keys we'll be registering meta with
```

```
luatest:setKeys({
```

```
  nwlanguagekey.create("test.alert"),
```

```
-- Step 3 - Define tokens that get you close to what you want
```

```
-- declare what tokens and events we want to match.
```

```
-- These do not have to be exact matches but just get you close to the data you want.
```

```
luatest:setCallbacks({
```

```
  ["cnn.com\013\010"] = luatest.tokenFIND,})
```

```
-- Step 4 - Do SOMETHING once your token matched
```

```
function luatest:tokenFIND(token, first, last)
```

```
  nw.createMeta(self.keys["test.alert"], "i_found_you")
```

```
end
```

# Appendix : Erroneous Parser Syntax

- ensure that if you manually edit file, that you could have your log decoder reload several times, and then it stops.
- If these errors occur, move out custom parser recently added and restart/start decoder service.
- **Example Events :**
  - Aug 22 14:33:42 testDecoder NwAppliance[4457]: [ServiceConnectionNode::messageHandler] [failure] localhost:50002: End of file
  - Aug 22 14:33:42 testDecoder init: nwlogdecoder main process (5185) killed by ABRT signal
  - Aug 22 14:33:42 testDecoder init: nwlogdecoder main process ended, respawning

# Appendix : XML Escaping Carrots

## Question:

How to escape carrot symbols “<” or “>”

Ex: &lt;username&gt;

## Answer :

Well, you cant use < or > directly in a content line.

You need to escape it.

&lt;&lt;&lt;username&gt;&gt;

Here &lt;&lt; escapes “<” and &gt; escapes “>”. And another paid is present to escape the variable itself.

# Appendix : Parser Deployment – new DIR create

Script for deploying new directories for parsers

```
for i in $(cat /root/scripts/applianceList/logDecoders); do
    ssh root@$i 'mkdir /etc/netwitness/ng/envision/etc/devices/PARSERDIR'
done
```

# Appendix : Parser Management – parser remove

```
#!/bin/bash
```

```
# Name : parserDeploy.sh
```

```
# Description : Deploys parsers on all log decoders
```

```
# Author : Joe Gumke
```

```
# Friendly Parser Name
```

```
parserDir=$1
```

```
# Raw Parser Name
```

```
#parser=$2
```

```
#parserPath="/etc/netwitness/ng/envision/etc/devices/$1/"
```

```
for i in `cat /root/scripts/applianceList/logDecoders`;
```

```
do
```

```
ssh root@$i "rm -rf /etc/netwitness/ng/envision/etc/devices/PARSERDIR/"
```

```
done
```

# Appendix : Parser Deployment – ini/xml deploy

```
# Name : parserDeploy.sh
# Description : Deploys parsers on all log decoders
# Author : Joe Gumke
# Example Usage : winevent_nic /tmp/v20_winevent_nicmsg.xml
# Example Usage : DIRECTORY_NAME PARSER_NAME
```

```
# Friendly Parser Name
parserDir=$1
# Raw Parser Name
parser=$2
parserPath="etc/netwitness/ng/envision/etc/devices/$1/"
for i in `cat /root/scripts/applianceList/logDecoders`;
do
    scp -p $2 root@$i:/$parserPath
done
```

# Appendix : Filetypespec deployment --

```
#!/bin/bash
# Name : fileTypeSpecDeploy.sh
# Description : Deploys parsers on all log decoders
# Author : Joe Gumke
# Example Usage : python fileTypeSpecDeploy.sh fileTypeSpecType FILENAME
# EXample Usage : python fileTypeSpecDeply.sh ODBC /tmp/hips8x.xml

# Friendly filetypespec Name
fileTypeSpecDir=$1
# Raw Parser Name
fileTypeSpec=$2
fileTypeSpecPath="etc/netwitness/ng/logcollection/content/collection/$1/"
for i in `cat /root/scripts/applianceList/logCollectors`;
do
    echo 'copying to:' $i
    scp -p $2 root@$i:$fileTypeSpecPath
done
```

# Appendix : Parser Basics-

- Eventcategory == legacy,-- NOT NEEDED-- every content message, developer uses taxonomy. Fixed value, cannot have multiple values. Has to be 10 digit number that is used in a mapping field to determine category mappings. Legacy to envision, and can omit this if you want.
- Functions Part --
  - @event\_time -- function -- what time the log came in.
- 
- Ip to device mapping -- new feature
- Support will check to allow to give the extra year information for logs that do not include that in the timestamp
- Every function should start with "@"



# Appendix : Identifying what header/message within the log parser matters

- In this case where you have 2 or more device with complete matches (HDR & MSG) for a single log the device that wins out is the first complete match according to alphabetical order. So, in this case since rhlinux (R) comes before symantecpgp (S) that match wins. If you change the name of the symantecpgp parser to say symantecpgp it will match. Or, use the IP to device map.
- Identification does not just stop at HDR analysis. It also uses MSG matching as well.
- So, you might have a strong HDR match but not a MSG match. Therefore a weak HDR match with a complete MSG match will win. In the case where you have a weak HDR match but not a MSG match then the strong HDR match with not a MSG match will win.

# Appendix : Windows events with separate parsers

- winevent\_nic
- winevent\_er?
- winevent\_snare > snare shops
- cef? >> nxlog?
- Parsers that identify from windows events : “%NICWIN-”
  - McAfeeVirusScan
  - MicrosoftIIS
  - mom
  - msexchange
  - msias
  - msisa
  - mssccm
  - mssharepoint
  - mssql
  - oracle
  - rsaacesrv
  - symantecav
  - vmware\_vc
  - avectopg
  - fsecureav
  - msforefront
  - netwitness

# Appendix : Device2ipmapping

- Helps build confidence in the ability for the log decoder engine to identify events to a specific parser (device.type).
- Helps confidence for wide/catchall parsers (rhlinux).
- Map IP to Device type in log parsing. Takes effect after parser reload.
- security.roles: parsers.manage
- parameters:
  - op - <string, {enum-one:edit|describe}> The operation to perform (edit|describe).
  - entries - <string, optional> The IP entries. StringParam in format of '+/-ip=device'. + means adding or editing a map entry, - means delete a map entry
  - reload - <bool, optional> If true, reload parsers when command finishes.
- Decoder > Explore > Parsers > ipdevice
- Op=describe
- 
- [http://sadoes.emc.com/0\\_en-us/090\\_10.4\\_User\\_Guide/120\\_ApplServConf/Decoder\\_and\\_Log\\_Decoder\\_Configuration\\_Guide/20\\_AddProc/MapIPtoServTyp](http://sadoes.emc.com/0_en-us/090_10.4_User_Guide/120_ApplServConf/Decoder_and_Log_Decoder_Configuration_Guide/20_AddProc/MapIPtoServTyp)

# Appendix : Parser Functions – url function

<@resultant:\*URL([ \$DOMAIN | \$ROOT | \$PAGE | \$QUERY | \$HOST | \$FQDN | \$PATH | \$EXT | \$PORT ],inputparam)>

Examples:

<@web\_domain:\*URL(\$DOMAIN,url)>

<@web\_root:\*URL(\$ROOT,url)>

<@web\_page:\*URL(\$PAGE,url)>

<@web\_query:\*URL(\$QUERY,url)>

For the following example:

inputparam = <http://hostname.subdomain.domain.topleveldomain:443/path/webpage.extension/?query=query>

Option	Resultant
\$DOMAIN	subdomain.domain.topleveldomain
\$ROOT	http
\$PAGE	webpage
\$QUERY	query
\$HOST	hostname
\$FQDN	hostname.subdomain.domain.topleveldomain
\$PATH	/path/
\$EXT	.extension
\$PORT	443

# Appendix : Parser Basics

- Content -- matches the token in the header to match to see static text and then the variables to match the text.
- Whole purpose of the header is to determine the message id
- Payload means what do you want to send to the message
- If there is content within the header that you want in the message parser, you payload REWIND.  
Not good practice to call out variables in the header parser



A nighttime city skyline is visible in the background, with several tall buildings illuminated. The scene is overlaid with a digital aesthetic, featuring a grid of binary code (0s and 1s) and circuit-like patterns in shades of blue and white. The text 'RSA Charge 2016' is prominently displayed in the center, with 'RSA' in a bold, white, sans-serif font, 'Charge' in a white, cursive script, and '2016' in a white, sans-serif font. The text is set against a glowing red rectangular background.

# RSA<sup>®</sup> Charge 2016

#RSACharge